FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO "EURÍPIDES DE MARÍLIA" – UNIVEM
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANDREI MITSUO DE BARROS NACAGAWA

TUTORIAL ORACLE FORMS BUILDER

MARÍLIA – SP

2006

ANDREI MITSUO DE BARROS NACAGAWA

TUTORIAL ORACLE FORMS BUILDER

Monografia apresentada ao Conselho do Curso de Bacharelado em Ciência da Computação do Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha, para obtenção do título de Bacharel em Ciência da Computação.

Orientador:
Prof. Dr. Edmundo Sergio Spoto

MARÍLIA – SP

2006

ANDREI MITSUO DE BARROS NACAGAWA

TUTORIAL ORACLE FORMS BUILDER

Banca examinadora da monografia apresentada ao Conselho do Curso de Bacharelado em Ciência da Computação do Centro Universitário Eurípides de Marília, para obtenção do título de Bacharel em Ciência da Computação.

Resultado:

ORIENTADOR: Prof. Dr. Edmundo Sergio Spoto

1° EXAMINADOR: Prof. Dr. Valter Vieira de Camargo

2° EXAMINADOR: Prof. Dra. Fátima L. Santos Nunes Marques

Marília, 04 de Dezembro de 2006.

Dedico este trabalho a todos aqueles que me ajudaram para a elaboração do mesmo.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus, que me forneceu todas as condições para que esse trabalho fosse desenvolvido.

Agradeço minha família, que sempre me deu forças nos momentos difíceis, durante esse período de elaboração do trabalho.

À minha noiva, grande amor da minha vida, presente em todos os momentos, sempre me apoiando e ajudando.

Aos meus amigos, que sempre me chamaram para tomar uma cerveja para relaxar, descontrair e jogar conversa fora.

Agradeço ao Dino (Professor e Orientador), que me conduziu no caminho certo para que eu pudesse estar aqui a um passo do meu objetivo.

Obrigado Haidie pelo resumo em Inglês e muito obrigado Jader pelo resumo em Espanhol.

Obrigado Arthur e Wendell, pelos materiais e dúvidas tiradas.

Valeu Rafa, pela ajuda na formatação do texto e das figuras.

Agradeço também à minha irmã Cissa, pela força e ajuda nos vários momentos que precisei.

Muito Obrigado a Todos.

NACAGAWA, Andrei M. B. **Tutorial Oracle Forms Builder**. 2006 – 83 fls. Monografia - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2006.

RESUMO

O Oracle Forms Builder é uma interface de desenvolvimento de aplicações cliente-servidor e web, que permite a geração de aplicações avançadas utilizando um banco de dados Oracle. Este tutorial mostra um estudo de caso de uma empresa de sacolas e fornece o Diagrama Entidade-Relacionamento e o script de geração das tabelas. A partir disso, foram desenvolvidos os aplicativos utilizando as técnicas e fundamentos da ferramenta, e demonstrados passo a passo como fazê-los. Através deste tutorial, toda pessoa da área de informática pode utilizar as técnicas aqui usadas e desenvolver seu próprio sistema.

Palavras-chave: Oracle Forms Builder. Formulário. Bloco de dados. Tabela.

NACAGAWA, Andrei M. B. **Tutorial Oracle Forms Builder**. 2006 - 83 fls. Monografia - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2006.

ABSTRACT

The Oracle Forms Builder is an interface of development of applications client-server and web, that allows the generation of advanced applications using a data base Oracle. This tutorial shows a study of case of a bag company and supplies to the Entity-Relationship Diagram and script of generation of tables. According to this, it was developed the applicatory using the techniques and fundamental of the tool, and we demonstrated step by step how to do them. Through this tutorial, every person of the computer science area can use the techniques used here and develop his own system.

Keywords: Oracle Forms Builder. Form. Data blocks. Table.

NACAGAWA, Andrei M. B. **Tutorial Oracle Forms Builder**. 2006. 83 fls. Monografia - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2006.

RESUMEN

El Oracle Forms Builder és una interface de desarollo de aplicación cliente-servidor e web, que permite la generación de aplicaciones avanzadas haciendo uso de un banco de datos Oracle. Este tutorial expone um estudio de caso de una empresa de bolsas y provee el Diagrama Entidad-Relacionamiento y los scripts de generación de las tablas. Para eso, hubo sido desarollado los aplicativos utilizando las técnicas y fundamentos de la herramienta y demostrados paso por paso como hacerlos. Por medio de este tutorial, cualquier persona de la área de informática puede hacer uso de las técnicas aquí utilizadas e desarollar su propio sistema.

Palabras-clave: Oracle Forms Builder. Formulario. Bloco de datos. Tabla.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SQL: Structured Query Language

PL/SQL: Procedural Language/Structured Query Language

WIMP: Windows, Icons, Menus, Pointer Devices

DML: Data Definition Language

DDL: Data Manipulation Language

LOV: List of Values

SUMÁRIO

INTRODUÇÃO

O Oracle Forms Builder é uma interface de desenvolvimento de aplicações cliente-servidor e *web*, que permite a geração de aplicações avançadas utilizando um banco de dados Oracle.

Os desenvolvedores de aplicativos fazem o projeto da entrada de dados e formulários de consulta com o Oracle Forms Builder; e os usuários finais podem então usar esses formulários para manipular os dados no banco de dados Oracle. A interface para o usuário é dirigida por eventos, ou por teclas de função ou teclado numérico.

O Oracle Forms Builder oferece uma estrutura de aplicação de ponta a ponta que inclui um conjunto integrado de construtores, assistentes de re-entrada, visualizadores ao vivo e paletas de propriedades. Os desenvolvedores conseguem criar gráficos, formulários de bancos de dados sofisticados e altamente interativos, além de lógica corporativa com mínimo esforço.

Uma das vantagens do Oracle Forms Builder é que as aplicações desenvolvidas através dele fazem a comunicação com o banco de dados sem que sejam precisos códigos extensos para isso.

A versão mais atual do Oracle Forms Builder é a 9i. Neste tutorial, optou-se por utilizar a versão 6 porque as duas versões são praticamente iguais para o desenvolvimento de aplicativos, sendo que a Versão 6 necessita de máquinas muito menos potentes e é fácil de configurar, devido ao fato de ser baseada no conceito de arquitetura em duas camadas e não exigindo a configuração de um servidor de aplicativos. A grande vantagem da Versão 9i é que por utilizar a arquitetura multi-camadas, seus aplicativos são carregados em um *browser* da *Web*. Ao final do capítulo 2 é dada uma explicação mais aprofundada do modo como a versão 9i funciona.

O tutorial em Oracle Forms Builder proposto neste trabalho é uma alternativa de aprendizado às pessoas interessadas em conhecer essa ferramenta muito poderosa para o desenvolvimento de sistemas integrados com o banco de dados. Nele são apresentados os principais conceitos de interface, Linguagem SQL e da própria ferramenta.

Para o entendimento dos passos do tutorial, foi utilizado como exemplo um sistema de uma empresa de embalagens, e disponibilizados os casos de uso e o script da criação das tabelas necessárias para o projeto.

Objetivos

O presente trabalho tem o intuito de criar um tutorial da ferramenta Oracle Forms 6i, mostrando desde a criação das tabelas no banco de dados Oracle até o desenvolvimento de aplicações que façam interações com essas tabelas.

Pretende-se assim familiarizar os leitores com os formulários de desenvolvimento de processos, para que a partir de então possam gerar suas próprias aplicações empregando os conceitos e metodologias aqui abordados.

Motivação

O Oracle Forms é uma ferramenta muito poderosa para o desenvolvimento de aplicações. Mesmo assim, há uma carência de material prático que ensine iniciantes a desenvolver seus próprios projetos. É com o intuito de preencher essa carência que foi desenvolvido este tutorial.

Assim, desenvolvedores que desejam ingressar no conhecimento dessa ferramenta podem acompanhar passo a passo o desenvolvimento de um sistema de Banco de Dados Relacional utilizando técnicas de interface.

Estrutura do Trabalho

No Capítulo 1 são apresentados os principais conceitos de interação, PL/SQL. Em seguida, no Capítulo 2, são apresentados os conceitos e componentes do Oracle Forms Builder. No Capítulo 3 é dada uma visão geral de todo o sistema, apresentando os casos de uso e o diagrama entidade-relacionamento. E, finalizando, no Capítulo 4, é mostrado o tutorial do Oracle Forms Builder.

1. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é feita uma introdução teórica aos principais conceitos relacionados ao Oracle Forms Builder. Inicialmente são apresentadas as definições referentes à interação. Em seguida, são mostrados os principais conceitos básicos em PL/SQL.

1.1. Interação

Um sistema interativo, ilustrado na Figura 1.1, possui o componente aplicação e o componente interface. O componente aplicação é o elemento responsável pela parte funcional do sistema, ou seja, que transforma dados de entrada em dados de saída através da aplicação de uma função à entrada. “O componente interface homem-máquina é o elemento responsável por traduzir ações do usuário em ativações das funcionalidades da aplicação, permitir que os resultados possam ser observados e coordenar esta interação” (LUCENA, 1998).

Em outras palavras, a interface é responsável pelo mapeamento das ações do usuário sobre dispositivos de entrada em pedido de processamento fornecido pela aplicação, e pela apresentação dos resultados produzidos na forma adequada.
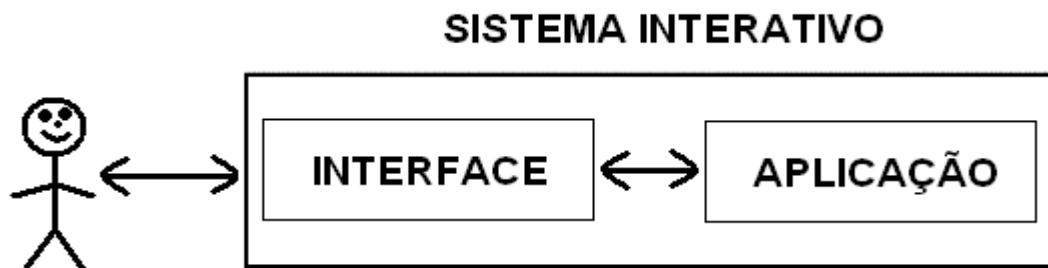
Figura 1.1 - Sistema Interativo

"Desde o início da computação, até os dias atuais, as interfaces podem ser classificadas em 4 gerações de estilo" (SILVA, 1997).

A primeira geração,nos anos 50 a 60, fazia uso de cartões perfurados, sendo que quase não existia a interface com o usuário. A comunicação era puramente textual e impulsionada por comandos e respostas a consultas geradas pelo sistema.

Em seguida, anos 60 a 80, surgiu a segunda geração de interfaces, cuja interação dava-se através de um monitor monocromático e um teclado alfanumérico. Nesse monitor, eram mostradas as listas de opções de comandos onde o usuário digitava o código do comando a ser executado.

A partir dos anos 80 até os dias de hoje, predomina a terceira geração de interface. Essa interface, chamada de WIMP (*Windows*, *Icons*, *Menus*, *Pointer Devices*)¸ oferece ao usuário múltiplos canais de entrada de dados para acesso e controle de múltiplas janelas. O uso de ícones gráficos, menus *pull-down*, botões e técnicas de *scrolling* reduzem a quantidade de digitação. Isso aumenta a eficiência de interação dos que não são datilógrafos experientes e torna o computador acessível a usuários que têm fobia de teclado.

Quando observada a evolução dessas três últimas gerações de interfaces, fica claro o uso que vem sendo feito da tecnologia para tornar mais fácil e natural a interação entre usuário e computador. Como continuação dessa tendência, tem-se o surgimento de uma quarta geração de interface batizada por alguns autores (DAM, 1997) como interfaces pós-WIMP. Como principais características dessas interfaces, destacam-se: a capacidade de reconhecimento da fala e, principalmente, o emprego de técnicas de Realidade Virtual.

Os estilos de interação, ou modo de diálogo, são a forma com que o sistema interage com o usuário. Em geral, vários estilos estão presentes em uma mesma interface. Os estilos comumente usados são: seleção de menus, linguagem de comandos, linguagem natural e caixa de diálogo.

1.1.1. Seleção de Menus

Na interação através de menus, a cada passo são apresentadas ao usuário as opções disponíveis. Assim há uma queda na quantidade de digitação, e se reduz o treinamento e o esforço de memorização necessários, já que o usuário precisa apenas relembrar a função associada a cada item.

“O sistema de menu consiste em um conjunto de alternativas apresentadas ao usuário, da mesma forma como se ele estivesse selecionando um prato no cardápio de um restaurante” (LUCENA, 1998).

Os menus podem ser classificados em dinâmicos e estáticos. Os menus estáticos devem ser mostrados apenas para: usuários novatos; usuários experientes que não estão familiarizados com algumas partes do sistema; usuários ocasionais que precisam relembrar alguma função; ou quando solicitados. A quantidade de espaço dos menus estáticos na tela deve ser minimizada.

Os menus dinâmicos são mais úteis para usuários de todos os níveis de experiência porque seus conteúdos são imprevisíveis e eles podem fornecer um retorno sobre resultados de operações.

Segundo Lucena (1999) os menus podem ainda apresentar uma outra classificação conforme suas escolhas.

Menu binário apresenta escolhas do tipo "Sim/Não". No Oracle Forms, esse tipo de menu geralmente é usado para mostrar alertas ao usuário, como é apresentado na Figura 1.2.
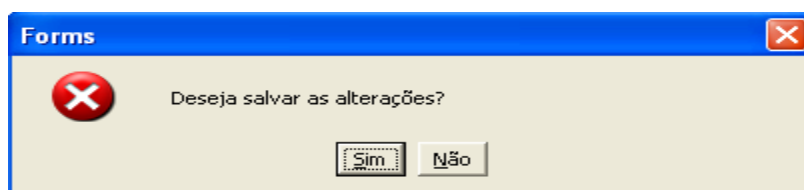
Figura 1.2- Menu Binário – Mensagem de alerta do Oracle Forms

Menu permanente, ilustrado na Figura 1.3, aparece constantemente, permitindo, a qualquer momento, uma seleção do item desejado.

Figura 1.3 - Menu Permanente – Barra de Menu do Oracle Forms

Menu *pull-down* ou *pop-up,* ilustrado na Figura 1.4, aparece na tela em conseqüência de um pedido por parte do usuários; o que ocorre é a superposição ou desmembramento de uma nova tela à que estava em uso, contendo novos itens de opção ou informação.
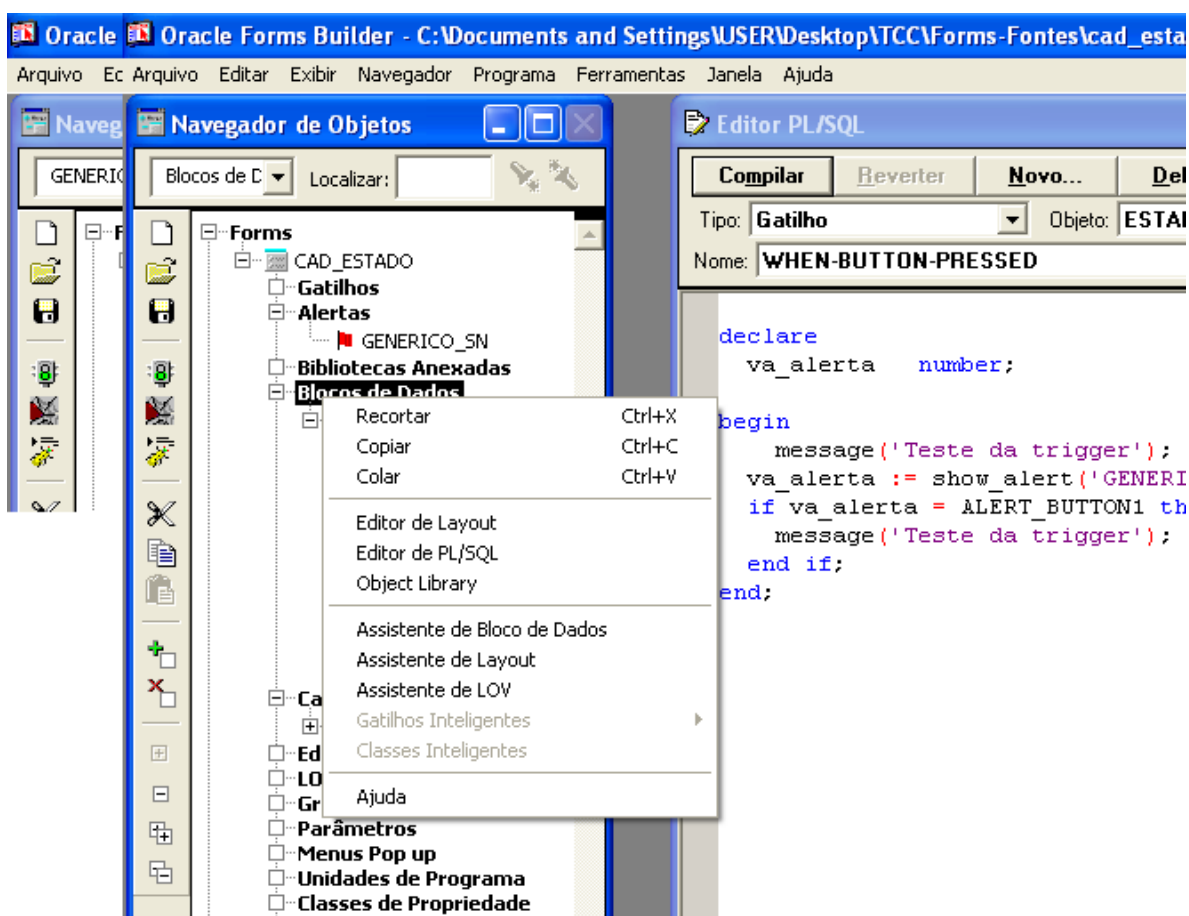
Figura 1.4 - Menu *Pull-down* ou *Pop-up*

"A apropriada organização da estrutura de menus é fundamental para a redução do tempo de entrada. Os itens do menu podem ser organizados em um menu simples, em uma seqüência linear de menus, em árvore, ou em rede" (LOH, 1998).

No menu simples todas as opções estão inseridas em um só menu, indicado para sistemas pequenos, como no caso do sistema que será desenvolvido. No menu de seqüência linear é exibida uma seqüência de menus (independente das opções do usuário). No menu em árvore, conforme a escolha do usuário, é apresentado um outro menu. No menu em rede para sistemas que provêm caminhos entre seções distintas. Como exemplo, para esclarecer melhor, constitui-se nos sistemas hipertexto, conforme ilustrado na Figura 1.5.
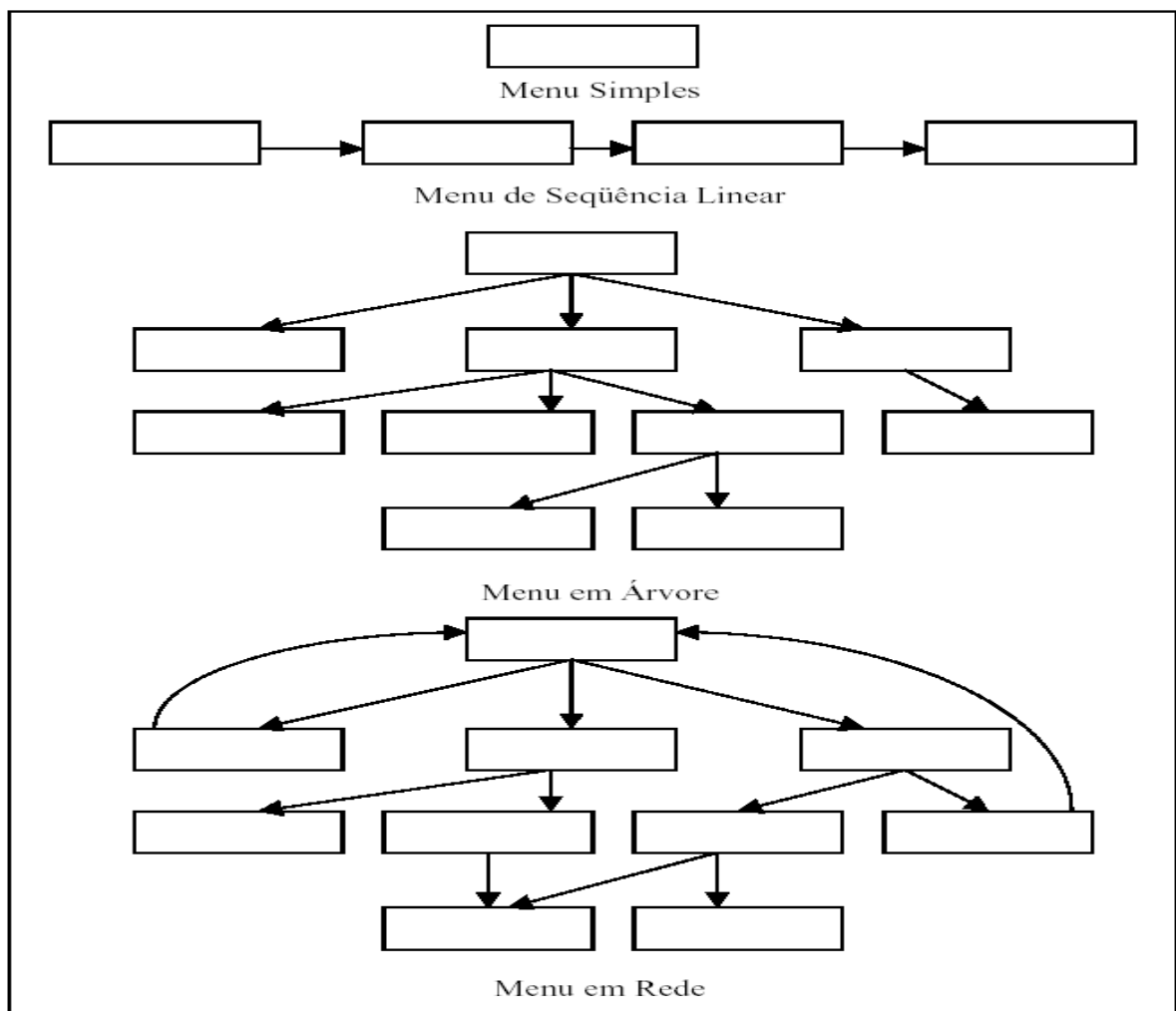
Figura 1.5 - Possíveis organizações dos menus

Algumas vantagens dos menus apresentadas por (SHNEIDERMAN, 1983) são: redução de digitação; redução de tempo de aprendizagem; torna as decisões estruturadas; permite o uso de ferramentas de gerenciamento de diálogos e permite fácil suporte para manuseio de erros. Mas também são citadas algumas desvantagens como: apresentar o perigo de muitos menus; pode ser devagar para usuário freqüentes; consome espaço de tela e requer *display* rápido.

Segundo Frainer (1993), o número de itens (amplitude) e o número de níveis (profundidade) em um menu devem ser observados. Estudos comprovam que cada menu deve ter de 4 a 8 itens, no máximo, e não mais que 3 ou 4 níveis. O aumento da profundidade diminui a velocidade e a precisão, uma vez que são necessárias mais seleções, e o aumento de itens dificulta a gerência do usuário, que deve fazer a busca pelo item desejado em um conjunto maior.

Os itens em um menu devem estar dispostos de forma natural. Não havendo uma seqüência óbvia, uma alternativa pode ser adotada: alfabética, freqüência de utilização ou importância.

1.1.2. Sistemas de Janelas

O sistema de janelas é um dos mais sofisticados meios de comunicação homem/máquina. É possível "abrir janelas" em vários locais da tela e colocar, funcionando em cada uma delas, uma parte do *software* aplicativo. A abertura simultânea de janelas múltiplas aproveita o pequeno espaço oferecido pelo monitor ou vídeo da máquina. Deste modo, pode-se tratar de vários aspectos do problema ou obter informações ao mesmo tempo, sem perder o controle e contato com as demais partes. Tem-se um exemplo de janelas na

Figura 1.6, onde é mostrada a página principal do Oracle Forms e suas principais ferramentas: Navegador de Objetos; Editor de Layout; e Paleta de Propriedades.
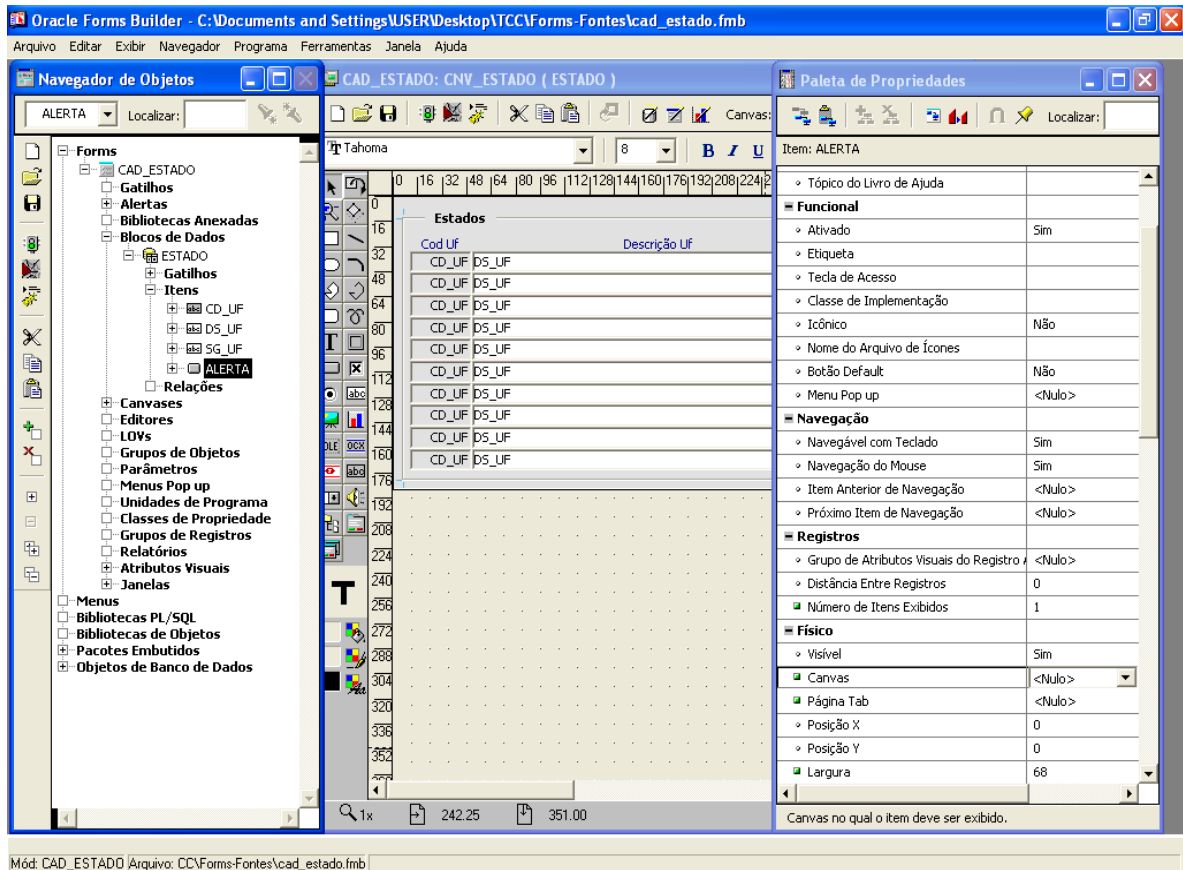
Figura 1.6 – Sistemas de janelas

1.1.3. Caixa de Diálogo

As caixas de diálogo são apresentadas ao usuário como constituídas por campos. A cada campo está associado um nome e um tipo, que determina o comportamento do campo, ou o modo que se dará a entrada.

Conforme Frainer (1993), os campos associados à caixa de diálogo são os apresentados na Tabela 1.1. A presença dos campos dependerá da função da caixa de diálogo e das opções que ela apresentará.

Tabela 1.1 – Campos Associados a Caixa de Diálogo

Campo	Descrição
Editor	Oferece um editor, com toda a funcionalidade necessária, para que o usuário submeta dados. Figura 1.7.
Botão de Pressão	Constitui-se de um botão que pode ser pressionado através da tecla "*enter*" ou através do clique do mouse. Figura 1.8.
Lista de Seleção Única	Consiste de uma lista na qual o usuário pode selecionar um item, deslocando o cursor com as teclas de setas e pressionando a tecla "*enter*", ou com um clique do mouse. Figura 1.9
Lista de Seleção Múltipla	Permite ao usuário selecionar vários itens da lista, deslocando o cursor com as teclas de setas e marcando os itens desejados através da pressão da barra de espaço, ou utilizando o mouse. Figura 1.10.
Seleção de Item	Permite que o usuário selecione opções fixas. Figura 1.11.
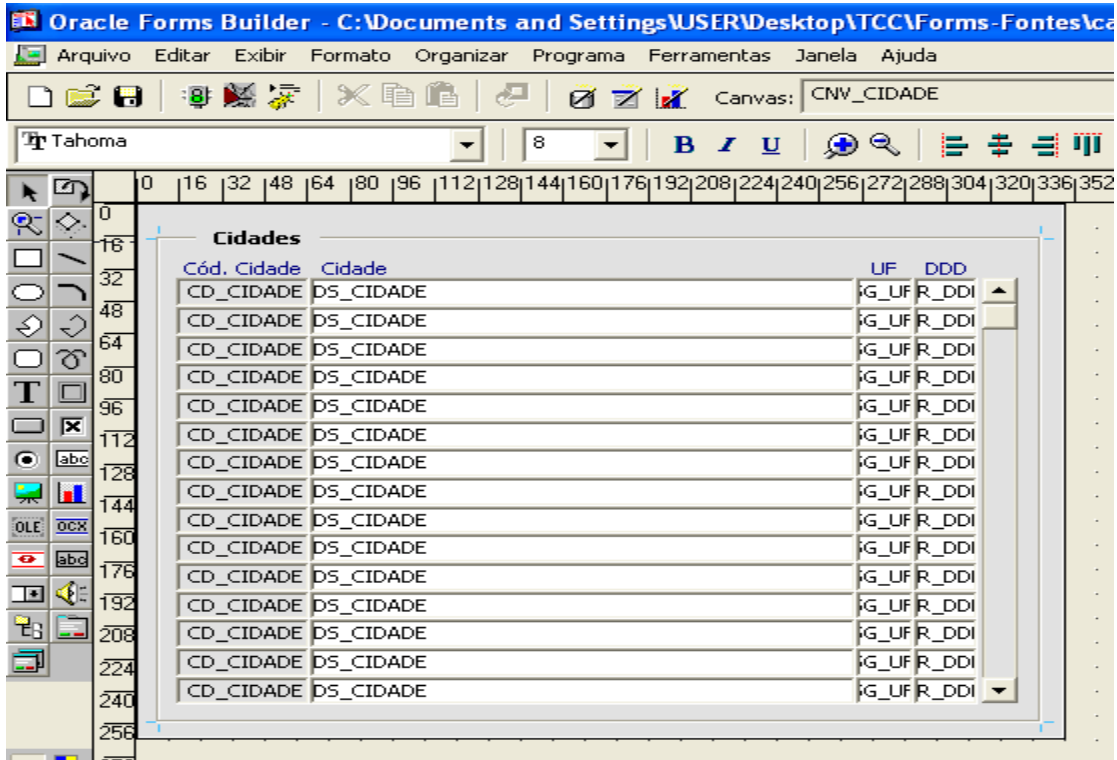
Figura 1.7 – Exemplo Caixa de Diálogo do tipo Editor
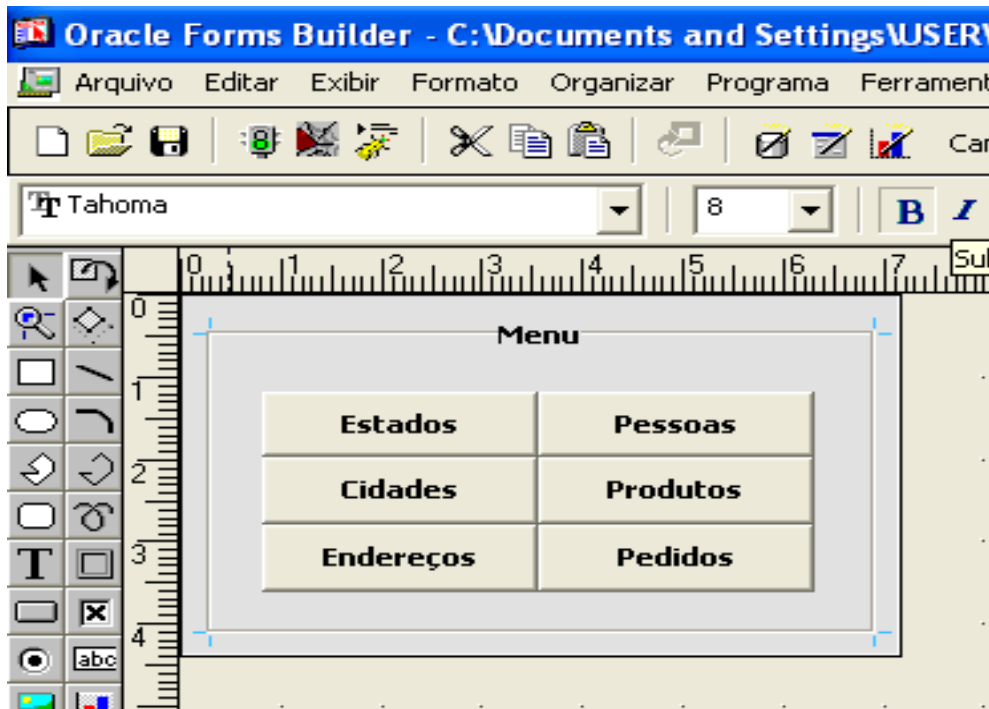
Figura 1.8 – Exemplo Caixa de Diálogo do tipo Botão de Pressão
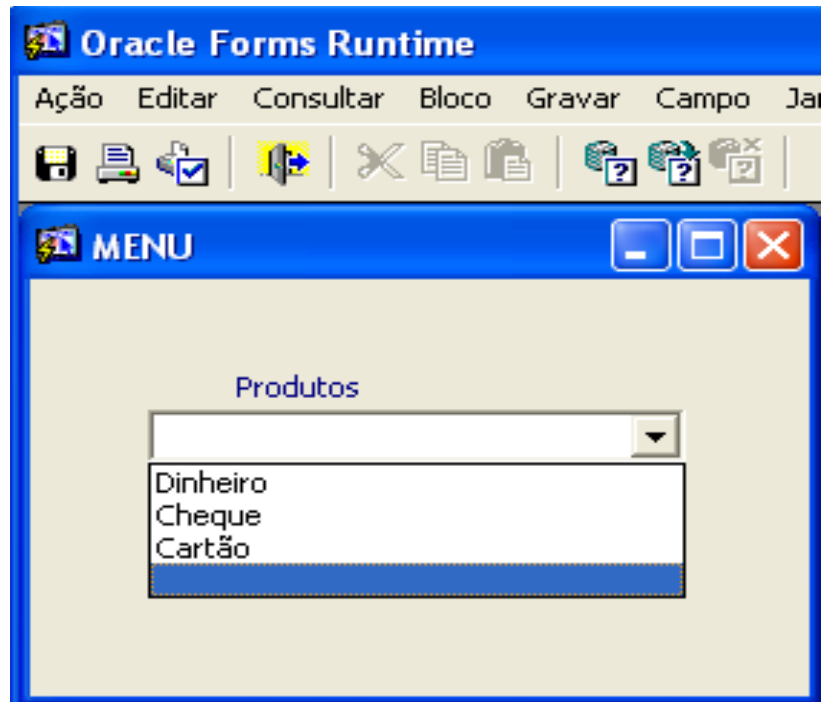
Figura 1.9 – Exemplo Caixa de Diálogo do tipo Lista de Seleção Única
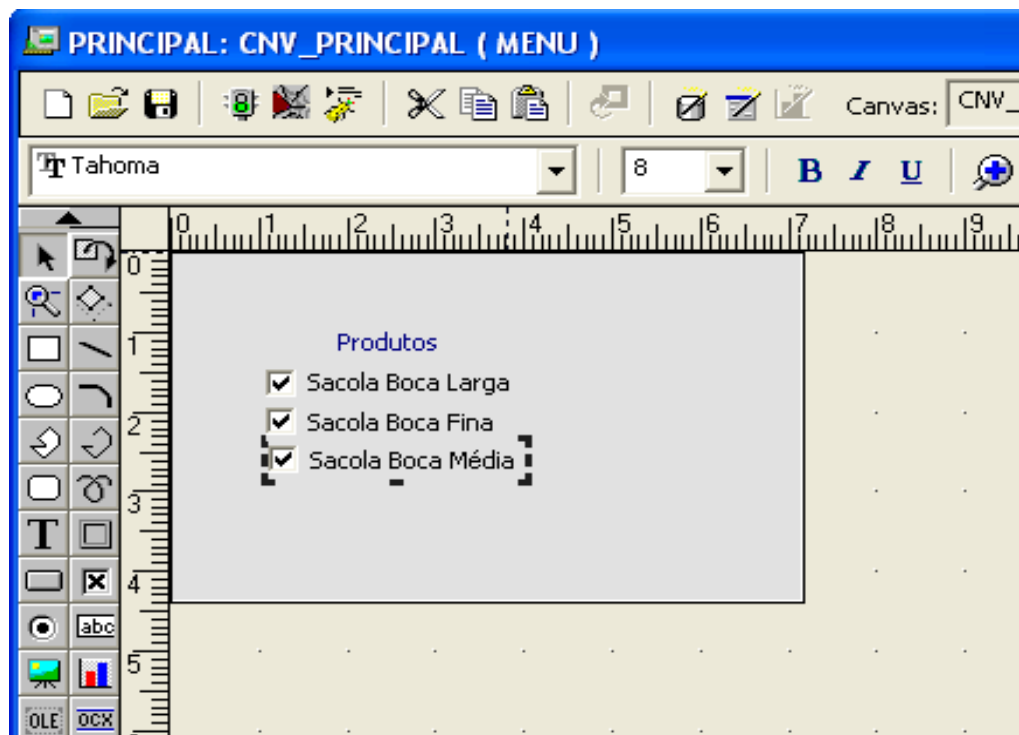
Figura 1.10 – Exemplo Caixa de Diálogo do tipo Lista de Seleção Única
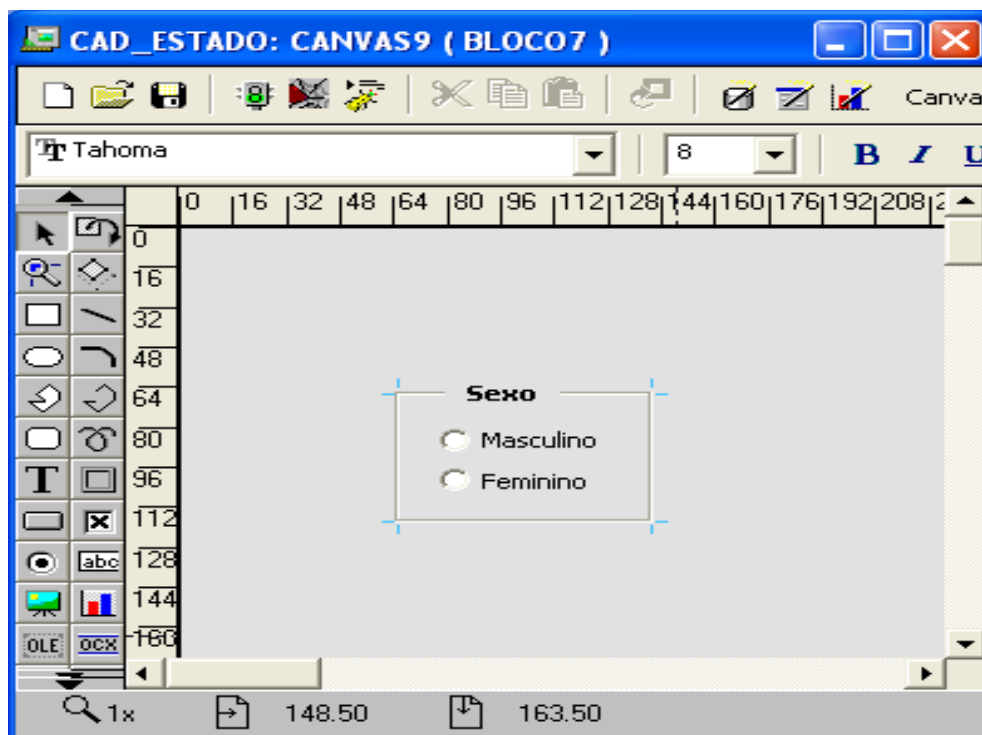
Figura 1.11 – Exemplo Caixa de Diálogo do tipo Seleção de Item

1.2. PL/SQL

"A linguagem de programação *Procedural Language/Structured Query Language* (PL/SQL) une o estilo modular de linguagens, como Pascal e C, à versatilidade no acesso a bancos de dados obtida via SQL" (MORELLI, 2002).

Segundo Pataballa (2001), o PL/SQL oferece recursos de engenharia de software modernos, como, por exemplo, para encapsular os dados, para tratamento de exceções, para a ocultação de informações, para a orientação de objeto e assim por diante, trazendo os recursos de programação mais modernos para o *Oracle Server* e o *Toolset*. O PL/SQL incorpora muitos recursos avançados feitos em linguagens de programação projetadas durante as décadas de 70 e 80. Além de aceitar a manipulação de dados, ele também permite que instruções de consulta do SQL sejam incluídas em unidades procedurais de código e

estruturadas em blocos, tornando PL/SQL uma linguagem de processamento de transações poderosa. Com o PL/SQL, pode-se usar as instruções SQL para refinar os dados da Oracle e as instruções de controle do PL/SQL para processar os dados.
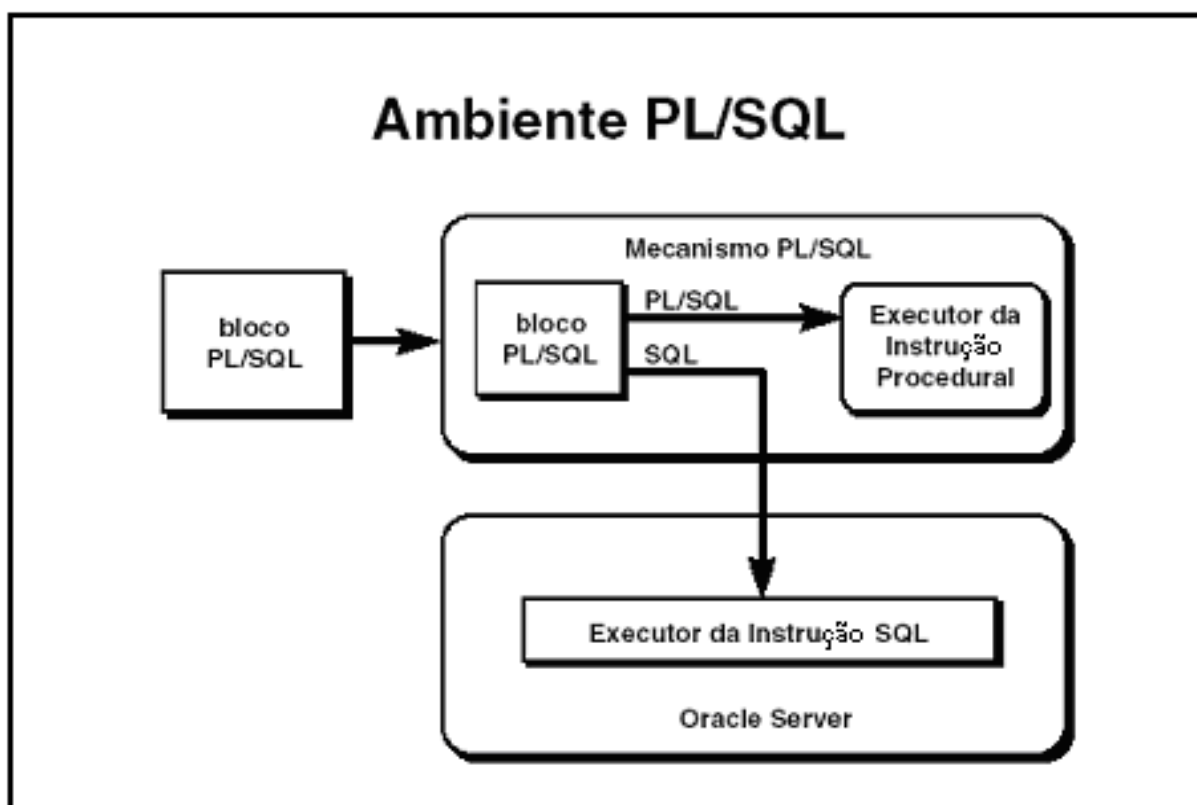
Figura 1.12 - Ambiente da Linguagem PL/SQL

O ambiente da Linguagem PL/SQL apresentada pela Figura 1.12 (retirada de PATABALLA, 2001), mostra como a Linguagem PL/SQL se comunica com o Executor de instrução do SQL *(Engine)* do SGBD Oracle.

Segundo Ramalho (1997), as vantagens de se usar PL/SQL são várias e normalmente associadas à performance e facilidade de programação, como :

- Redução do número de chamadas de uma aplicação para o servidor;
- Portabilidade entre as plataformas em que o Oracle é executado;
- Detecção e gerenciamento de erros.

“Usando PL/SQL, o servidor pode receber diversos comandos de uma só vez, reduzindo assim o tráfego entre a aplicação e o servidor além de reduzir o número de chamadas de I/O” (RAMALHO, 1997).

1.2.1. Características

As operações permitidas em PL/SQL são:

• Manipulação de Dados: alteração, eliminação, inclusão e seleção;

• Criar variáveis e constantes herdando o tipo de dados e o tamanho de outras variáveis e constantes ou de colunas de tabelas;

• Criar cursores para tratar o resultado de uma consulta que retorna 0 ou mais linhas;

• Criar registros para guardar o resultado de um cursor ou campos de tabelas, herdando os tipos de dados e tamanho das colunas;

• Tratar erros;

• Criar *labels* para controlar o fluxo de execução;

• Utilizar comando de repetição e comparação.

• Criar *triggers* (gatilhos) para garantia de integridade (restrições), segurança, etc.

1.2.2. Estrutura de um bloco PL/SQL

Segundo Ahuja (2002), a estrutura de um bloco PL/SQL é composta por uma área de declaração, uma área de comandos e uma área de exceções, como na Figura 1.13:

```
DECLARE (opcional)
        Declarações – variáveis, cursores, constantes, estruturas, tabelas
        exceções definidas pelo usuário
BEGIN (obrigatório)
        Estruturas executáveis (comandos)
        Instruções SQL (manipular dados do banco de dados)
        Instruções PL/SQL (manipular dados no bloco)
EXCEPTION (opcional)
        Tratamento de exceções (pode conter outros blocos)
        Ações a serem desempenhadas qdo ocorrem erros ou condições anormais
END; (obrigatório)
```

Figura 1.13 - Estrutura de um bloco PL/SQL

1.2.3. Tipos de Bloco

Blocos Anônimos: são blocos sem nome. Eles são declarados em um ponto do aplicativo onde eles devem ser executados e são passados para o mecanismo PL/SQL para serem executados em tempo de execução (ver Figura 1.14).

Subprogramas (procedimentos ou funções): são blocos PL/SQL nomeados que podem assumir parâmetros e podem ser chamados. Pode-se declará-los como procedimentos ou como funções, mostradas na Figura 1.14.
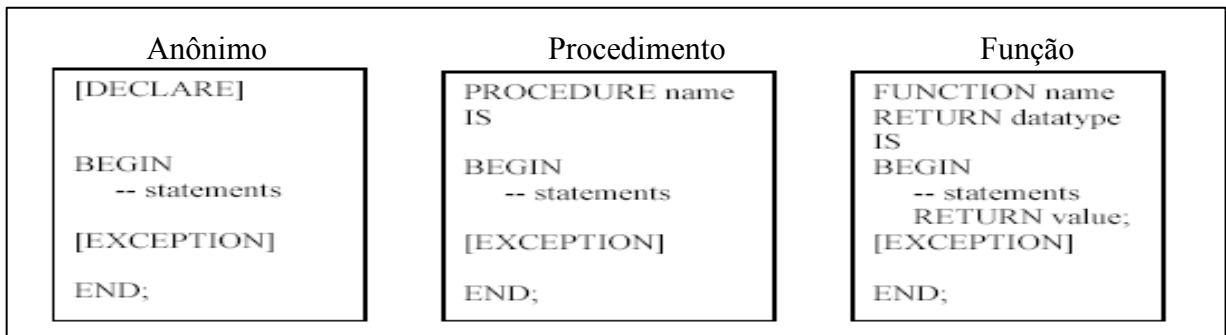
Figura 1.14 - Blocos de PL/SQL – Anônimo, Procedimento e Função

1.2.4. Tratamento de Exceções

"Uma exceção representa uma interrupção anormal no procedimento de uma *procedure* ou função" (MORELLI, 2002). Qualquer programa bem escrito tem de ter uma capacidade de tratamento de erros inteligente e com possível recuperação.

O PL/SQL implementa o tratamento de erros através de exceções e rotinas de tratamento de exceções. As exceções podem ser associadas a erros Oracle ou a erros definidos pelo usuário.

Quando ocorre um erro, é levantada uma exceção, e o controle é transferido para o bloco de tratamento de exceções. Diferentemente de outras linguagens, após a execução do bloco de tratamento de exceções, o bloco PL/SQL termina. Geralmente nesta instrução, dependendo da função do bloco, inclui- se um *rollback*.

A estrutura de um tratamento de exceção é demonstrada a seguir, na Figura 1.15.

```
DECLARE
        .....
BEGIN
        .....
EXCEPTION
        WHEN NO_DATA_FOUND THEN
                statement1;
                statement2;
        WHEN TOO_MANY_ROWS THEN
                statement1;
        WHEN NO_DATA_FOUND THEN
                statement1;
                statement2;
                statement3;
END;
```

Figura 1.15 – Estrutura de uma Exception

1.2.5. *Trigger* (gatilho)

"*Triggers* representam unidades de programa que são executadas, automaticamente, antes ou depois de um comando disparador, que pode ser tanto um DML, como um DDL ou mesmo um evento ocorrido no *Database* (uma conexão, por exemplo)" (MORELLI, 2002).

As *triggers* são semelhantes aos procedimentos e funções pelo fato de serem blocos de PL/SQL nomeados com seções declarativas, executáveis e de tratamento de exceções. Tal como os pacotes, as *triggers* têm de ser armazenadas na base de dados e não podem ser locais em relação a um bloco. No entanto, um procedimento é executado explicitamente a partir de outro bloco através de uma chamada de procedimento, que também pode passar argumentos. Uma *trigger* é executada implicitamente sempre que o acontecimento de *triggering* suceder e não aceita argumentos. O ato de executar uma *trigger* é conhecido como disparar a *trigger*.

O acontecimento de *triggering* é uma operação de DML (*INSERT*, *UPDATE* ou *DELETE*) numa tabela da base de dados.As *triggers* podem ser utilizadas para muitas operações, incluindo:

- Manter restrições de integridade complexas, que não são possíveis através de restrições declarativas ativadas na criação da tabela;
- Efetuar a auditoria às informações de uma tabela, registrando as alterações efetuadas a quem as efetuou;
- Indicar automaticamente a outros programas que é necessário efetuar uma ação, quando são efetuadas alterações numa tabela.

A estrutura de uma *trigger* é demonstrada a seguir, na Figura 1.16:

```
CREATE OR REPLACE TRIGGER nome_da_trigger {BEFORE / AFTER}
DELETE OR INSERT OR UPDATE OF (nome_coluna1, nome_coluna2, ...)
ON nome_da_tabela
REFERENCING [OLD AS antigo] [NEW AS novo]
FOR EACH ROW
WHEN condição
DECLARE
        Variáveis, constantes, etc.
BEGIN
        .....
        .....
END {não colocar o nome_da_trigger};
```

Figura 1.16 – Estrutura de uma *Trigger*

1.2.5.1. Tempo

O tempo da *trigger* indica quando a *trigger* será ativada em relação ao disparo do evento. Os tempos de uma *trigger* podem ser:

- *BEFORE* : antes do evento
- *AFTER* : depois do evento

1.2.5.2. Evento de Disparo

O evento de disparo indica qual a operação de manipulação de dados sobre uma tabela disparará a *trigger*. Os eventos de disparo podem ser:

• *INSERT*

• *UPDATE*

• *DELETE*

No caso de evento de disparo do tipo *UPDATE*, as colunas devem ser especificadas após a palavra *OF*.

1.2.5.3. Tipo

O tipo da *trigger* indica quantas vezes a *trigger* poderá ser disparada. Quanto ao tipo, uma *trigger* pode ser:

• Comando : manipula um grupo de dados dentro de uma tabela e executa uma única vez.

Exemplo: *DELETE* para um grupo de linhas.

• Linha : manipula linhas de uma tabela e pode ser executada uma ou mais vezes.

Exemplo: *INSERT* e *UPDATE* para linhas específicas.

Quando o disparo de um comando de manipulação de dados afeta múltiplas linhas, a *trigger* de comando dispara uma vez e a *trigger* de linha dispara uma vez para cada linha afetada pelo comando.

2. ORACLE FORMS BUILDER

"O Oracle Forms é uma ferramenta de construção de aplicativos cheia de recursos que produz telas de qualidade de produção utilizando dados armazenados em um banco de dados" (ABBEY et al., 1997).

Segundo Fernandes (2002), as características de uma aplicação Oracle Forms são as seguintes:

- Possibilita inserção, atualização, exclusão e consulta de dados usando uma variedade de interfaces gráficas;
- Apresenta dados utilizando-se dos itens tipo textos e imagens;
- Permite o desenvolvimento de aplicações utilizando janelas e transações de banco de dados;
- Possui integração com os produtos Oracle Report e Oracle Graphics, permitindo a utilização de recursos destes produtos;
- Apresenta-se sob forma de menus.

Segundo Fernandes (2002), o desenvolvedor de aplicativos pode:

- utilizar várias fontes de dados além dos bancos de dados Oracle;
- construir aplicações rapidamente utilizando as ferramentas gráficas;
- desenvolver aplicações portáveis para outras plataformas (inclusive em modo caracter);
- copiar ou mover objetos e suas propriedades entre os componentes das aplicações;
- utilizar a interface familiar a todos os componentes do *Developer*, como por exemplo, o *Layout Editor* e o *Object Navigator*;

O Oracle Forms possui três componentes:

- *Oracle Forms Generate* - interpreta as definições de um módulo, criando um arquivo executável.

- *Oracle Forms Runform* - executa um arquivo executável de uma aplicação Oracle Forms. Os arquivos utilizados no momento da execução já devem ter sido gerados pelo Oracle *Forms Generate*.
- *Oracle Forms Designer* - permite o design e armazenamento das especificações de um módulo tipo Form, Menu ou *Library*. Permite também invocar os outros componentes *Forms Generate* e *Forms Runform*.

2.1. Módulos do Oracle Forms

Uma aplicação Oracle Forms pode ser composta de muitos módulos. Estes módulos podem ser do tipo: *Form*, *Menu* ou *Library*.

2.1.1. Estrutura de um módulo Form

Módulos Form consistem no corpo principal de uma aplicação Oracle Forms. Apresentam os objetos e os dados que os usuários podem ver ou interagir. Os principais objetos de um módulo Form são:

- Itens - São objetos que apresentam valores de dados do usuário ou, dependendo do tipo de item, permitem ao operador interagir com o form. Itens estão logicamente agrupados em blocos visivelmente arranjados em *canvas*;
- Bloco - É o dono lógico de um item. Em um bloco os itens estão logicamente relacionados. Eles podem, por exemplo, pertencer a coluna de uma mesma tabela ou fazer parte de um mesmo ciclo de navegação.

- Canvas - É a superfície onde os objetos visuais estão arranjados (podendo imaginá-lo como a tela de um pintor). Um módulo Form pode ter várias *canvas*. Por default, todas os *canvas* ficam na mesma janela, mas podemos alocar janelas separadas para cada *canvas*.

A seguir, na Figura 2.1, tem-se a demonstração destes objetos.
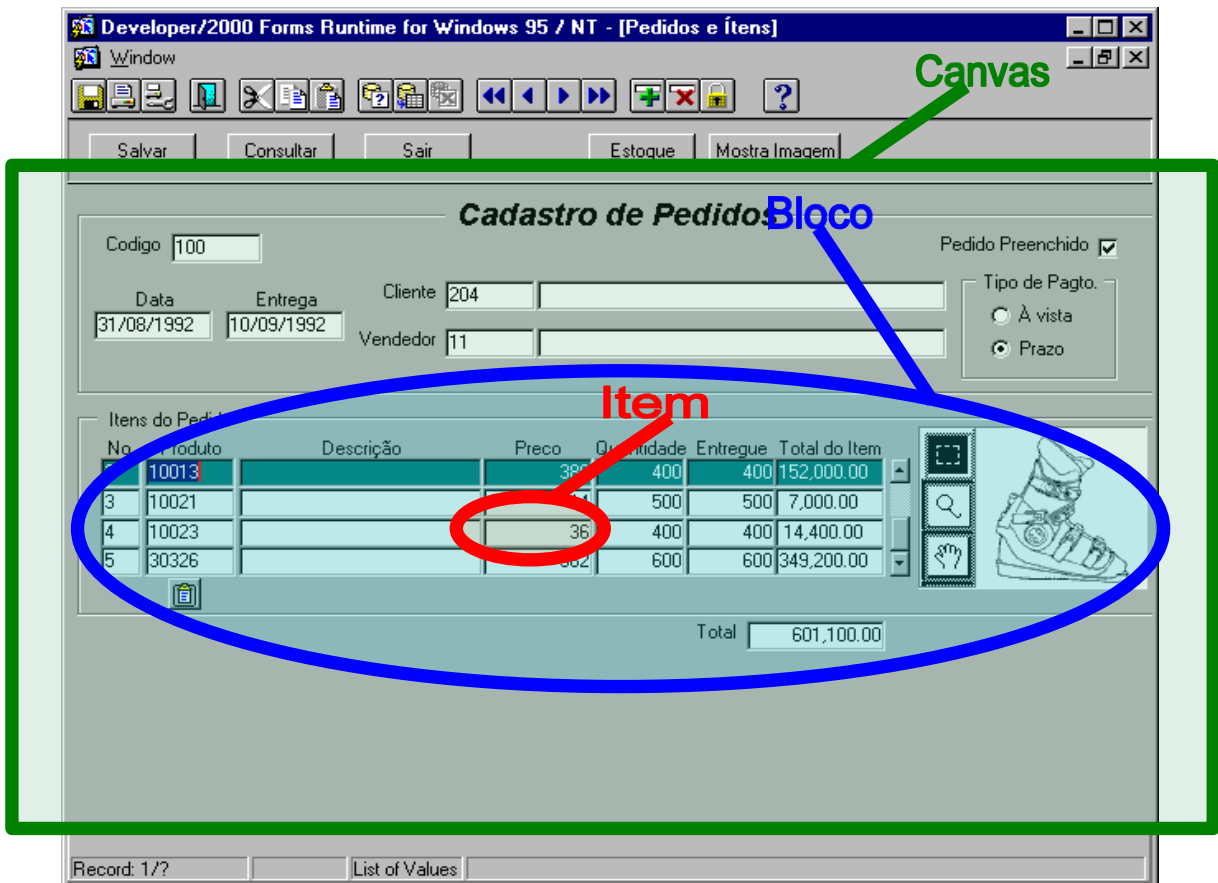
Figura 2.1 – Objetos do Oracle Forms

Os blocos podem ser classificados de três maneiras: bloco baseado em tabela, bloco baseado em *Select*, e bloco de controle.

O Bloco baseado em Tabela é aquele que está associado com uma tabela (ou *view*). Cada bloco pode ter somente uma tabela associada a ele, entretanto é possível acessar dados de outras tabelas, programando-se através de *triggers*. Na criação de um bloco baseado em tabela são automaticamente criados os itens para as consultas da tabela base;

O Bloco baseado em *Select* é um bloco baseado em um comando *Select*, que será executada para todas consultas efetuadas no bloco.

Já o Bloco de Controle é um bloco que não está associado a nenhuma tabela base. É usado para colocar botões que poderão ser colocados em um *Canvas*.

Para suportar o relacionamento entre blocos e suas tabelas base é possível definir um bloco como detalhe de um outro (mestre). É conhecido como Relacionamento Mestre-Detalhe. Isto permite valores de chave primária e de chave estrangeira ligados através de blocos, havendo desta forma sincronismo dos dados. O Oracle Forms automaticamente gera o código necessário para suportar o relacionamento mestre detalhe. Na Figura 2.2 tem-se o exemplo de um Relacionamento Mestre-Detalhe, onde a parte de cima do *canvas* contém o bloco Mestre, e a parte de baixo contém o bloco Detalhe.

Figura 2.2 – Exemplo de um Relacionamento Mestre-Detalhe

Nos blocos é possível definir quantos registros em bloco são mostrados por vez:

- Bloco *Single-record* Mostra um registro de cada vez.
- Bloco *Multi-record* Mostra mais de um registro de cada vez.

Os itens dos blocos podem ser de diferentes tipos, conforme já especificado na Seção 1.1.3.

2.1.2. Estrutura de um módulo Menu

Um módulo de menu é um módulo de estrutura hierárquica que provê um método rápido e fácil de operação para uma aplicação Oracle Forms. Módulos Menu são atachados a módulos Form e é um dos principais componentes de uma aplicação. Na parte superior da Figura 2.3 tem-se um exemplo de Menu:
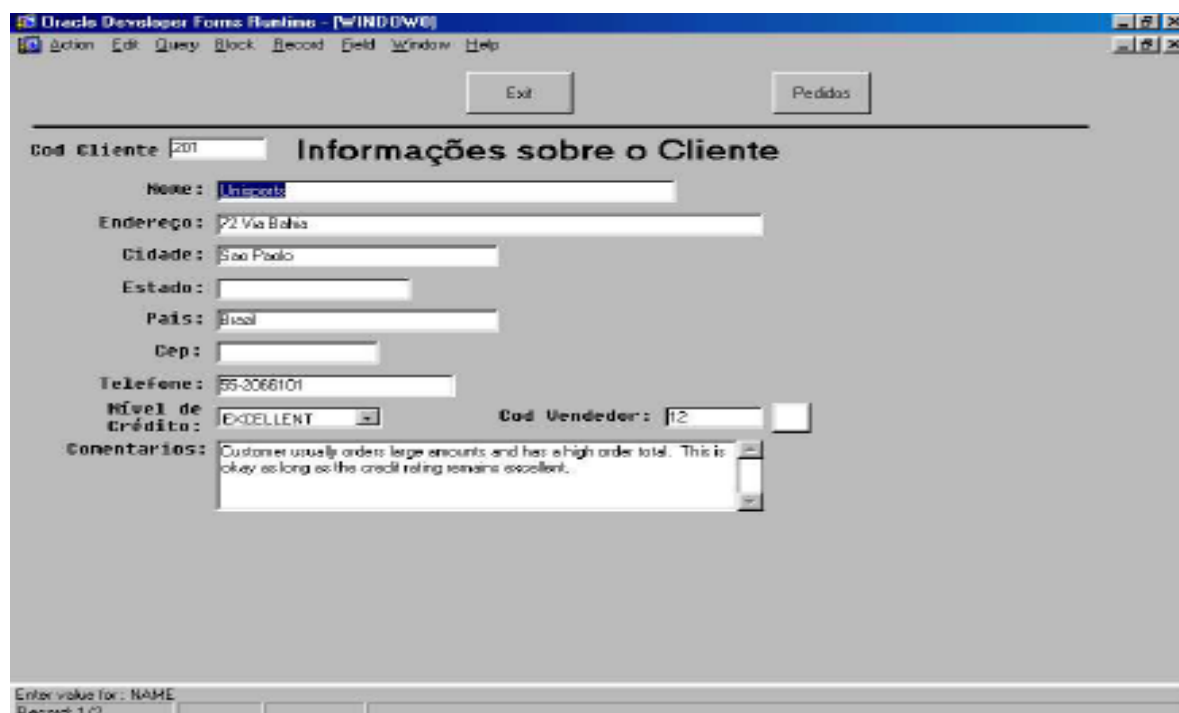
Figura 2.3 – Exemplo de um Módulo Menu

Um menu é uma lista de opções relacionadas. Cada opção executa uma ação diferente. Existem 3 tipos de menus:

- Main Menu: é o menu principal. Mostrado horizontalmente na barra de menu. Contém opções (Menu Itens) que são geralmente menus individuais;
- Individual Menu: é o menu individual, que é mostrado verticalmente.
- Sub-menu: mostrado verticalmente e à direita do Menu Item que o chamou.

2.1.3. Estrutura de um módulo Library

Módulo do tipo *Library* é um conjunto de sub-programas PL/SQL (como em outras ferramentas PL/SQL).

2.2. Tipos de consultas aos dados

Existem dois modos para consultar os dados:

- Incondicional: seleciona todas as linhas de todas as colunas representadas no bloco em relação a tabela base sem especificar condição.
- Condicional: seleciona somente as linhas que satisfaçam as condições especificadas para todas as colunas representadas no bloco em relação a tabela base.

2.3. Modos de Operação

O Oracle Forms possui dois modos de operação: modo Normal e modo *Enter Query*.

2.3.1. Modo Enter Query

É o modo de operação que permite a entrada de critérios de pesquisa para uma consulta.

No modo *Enter Query* é permitido:

- Consultar todos os registros;
- Consultar registros utilizando um critério;
- Consultar registros especificando-se uma cláusula *WHERE* mais complexa para uma consulta. Neste caso tem-se que utilizar a opção &.

No modo *Enter Query* não é permitido:

- Navegar fora do bloco corrente;
- Sair da sessão *Runform*;
- Usar certas funções, tais como *Next Record*;
- Inserir registros;
- Atualizar registros;
- Excluir registros.

Nos aplicativos criados no Oracle Forms alguns botões são disponibilizados para que sejam feitas as consultas. Na Figura 2.4 é mostrado o botão para entrar no modo *Enter Query*. Na Figura 2.5 é ilustrado o botão para executar a consulta e na Figura 2.6 é representado o botão de cancelamento da consulta.
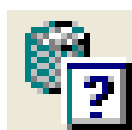
Figura 2.4 – Botão para entrar no Modo *Enter Query*
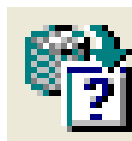
Figura 2.5 – Botão para Executar a Consulta

Figura 2.6 – Botão para Cancelar a Consulta

2.3.2. Modo Normal

É o modo de operação que permite a inserção, atualização e exclusão de registros (linhas) do Banco de dados.

No modo normal é permitido:

- Consultar registros;
- Inserir novos registros;
- Atualizar registros;
- Excluir registros;
- Efetivar as mudanças (*Commit*);
- Sair da sessão *Runform*.

No modo Normal não é permitido:

- Executar uma consulta condicional;
- Utilizar o diálogo *Query Where*.

Na Figura 2.7 é representado o botão destinado à inserção de novos registros nas aplicações. Já na Figura 2.8 é representado o botão de exclusão de registros.

Figura 2.7 – Botão para Inserir Registros

Figura 2.8 – Botão para Excluir Registros

2.4. LOVs - Listas de Valores

Uma LOV é um tipo de objeto utilizado para prover ao operador uma lista de valores dinâmica, LOVs são definidas a nível de formulário, ou seja podem ser utilizadas por mais de um item em qualquer bloco do módulo formulário.

Quando se cria uma LOV existem 3 tipos de objetos envolvidos:

1. Grupo de Registros - objeto utilizado para armazenar o *array* de valores representados pela lista. Pode ser criado antes ou fazer parte do processo de criação LOV.
2. LOV - lista de valores propriamente dita. Apresenta os valores, a partir do Grupo de Registros, que poderão ser selecionados e passados para os itens do form.
3. *Text itens* - Item tipo texto do form associado à lista de valores (LOV), a partir do qual a lista de valores poderá ser chamada.

2.5. Triggers

Em uma *trigger* existem três componentes necessários para sua construção: tipo de *trigger*, corpo da *trigger*, e escopo da *trigger*.

2.5.1. Tipo de Trigger

Caracteriza o evento que faz com que a *trigger* dispare. A primeira parte do nome (prefixo) segue um padrão que permite a identificação do tipo de *trigger*. Todas as *built-in trigger* são associadas a um evento e seus nomes sempre contêm um prefixo seguido de hífens (-). Os tipos são:

- *PRE-*: A *trigger* é disparada quando ocorre um evento antes de uma ação (exemplo: uma consulta é executada *PRE-QUERY*);
- *POST-*: A *trigger* é disparada quando ocorre um evento depois de uma ação (exemplo: uma consulta é executada *POST-QUERY*);
- *WHEN-*: A *trigger* é disparada em adição ao processamento padrão, permite aumentar a funcionalidade (exemplo: na validação de um item *WHEN-VALIDATE-ITEM*);

ON-: A *trigger* é disparada em substituição ao processamento padrão, permitindo o aumento da funcionalidade (por exemplo quando ocorrer um erro *ON-ERROR*).

2.5.2. Código da Trigger

É uma ação que a *trigger* executa quando disparada. Escreve um bloco PL/SQL anônimo utilizando o PL/SQL editor, onde se pode utilizar normalmente os recursos da Linguagem PL/SQL.

2.5.3. Escopo da Trigger

O escopo de uma *trigger* é determinado pela sua exposição na hierarquia dos objetos do *form*, ou seja, embaixo de que tipo de objeto foi criada a *trigger*. Os níveis são:

- Nível de Form: A *trigger* pertence ao módulo e é disparada quando o evento a qual ela está associada ocorrer em qualquer posição no *form* (qualquer bloco, qualquer item);
- Nível de Bloco: A *trigger* pertence ao bloco e somente é disparada quando o evento a qual está associada ocorrer neste bloco (em qualquer item dentro do bloco da *trigger*.);
- Nível de item: A *trigger* pertence ao item e somente é disparada quando o evento a qual ela está associada ocorrer neste item (específico).

O estilo de execução é a propriedade da *trigger* que controla o que acontece quando há mais de uma *trigger* do mesmo tipo em diferentes níveis. As definições para o estilo de execução são:

- *Override*: Somente a *trigger* mais específica é disparada;
- *After*: A *trigger* é disparada depois de disparada a *trigger* do mesmo tipo em nível mais alto (se existir);
- *Before*: A *trigger* é disparada antes de disparada a *trigger* do mesmo tipo em um nível mais alto (se existir).

2.6. Validação

O Oracle Forms executa o processo de validação em vários níveis para garantir que registros e valores individuais sigam determinadas regras. Níveis em que ocorrem as validações:

- Item: O Oracle Forms registra o status de cada item de modo a determinar se será válido. Se um item é alterado e ainda não está marcado como válido então o Oracle Forms primeiro executa a checagem de validação padrão conforme as propriedades do item. Estas checagens são executadas antes de disparar a *trigger WHEN-VALIDATE-ITEM*.
- *Record*: Depois de deixar o registro, o Oracle Forms verifica se o registro é válido e a *trigger WHEN-VALIDATE-RECORD* é disparada, se presente. Quando o registro passa estas checagens ele é alterado para válido.
- *Block* e Form: Todos os registros abaixo deste nível são validados.Por exemplo, ao *COMMIT* do *form*, então todos os registros do *form* são validados, a menos que esta ação seja suprimida.

2.7. Versões do Oracle Forms Builder

A principal diferença entre as duas últimas versões do Oracle Forms Builder é que a versão 6 é baseada na arquitetura de duas camadas e a versão 9i na arquitetura multi-camadas. Sendo assim, na versão 9i é necessária a configuração do servidor de aplicativos.

Segundo Rodrigues (2002), as aplicações baseadas na arquitetura duas camadas, conhecidas também como Cliente/Servidor, funcionam dividindo a aplicação nas estações e no servidor de dados. Na primeira camada (cliente), onde se encontra a apresentação, as

regras de negócios e o acesso aos dados, é feita a requisição de informações que constam no servidor. Assim, se conectam à segunda camada (servidor), onde há ligação com o SGBD (Sistema Gerenciador de Banco de Dados) que serve aos clientes o que foi solicitado. Toda ligação entra a máquina cliente e a servidora é feita diretamente e através de protocolos de comunicação (ex. TCP/IP).

Ainda segundo Rodrigues (2002), as aplicações desenvolvidas na arquitetura multi-camadas são a evolução do modelo cliente/servidor, onde surge uma (ou mais) camada intermediária entre a parte cliente e o servidor de dados.

“O ponto fundamental desse modelo é a divisão distinta entre as camadas onde não há a intromissão de uma na outra. A divisão é feita entre a Camada de Apresentação, Camada de Negócios e Camada de Dados” (RODRIGUES, 2002).

A Camada de Negócios é conhecida também como Servidor de Aplicação e é o intermediário entre o cliente e o servidor, sendo ela responsável por receber as requisições do cliente e fazer o acesso aos dados no servidor. É onde a aplicação pode se tornar 3, 4 ou n camadas, bastando acrescentar o número de camadas necessárias.

Na Figura 2.9, (retirada de BALA, 2006) é demonstrado como é o funcionamento da versão 9i do Oracle Forms Builder. A camada do cliente contém o *browser* da *Web* onde os aplicativos são carregados. A camada do meio é o servidor de aplicativos onde as aplicações lógicas e o *software* servidor estão. Na camada do Banco de Dados está o servidor do Banco de Dados onde os dados da empresa são armazenados.
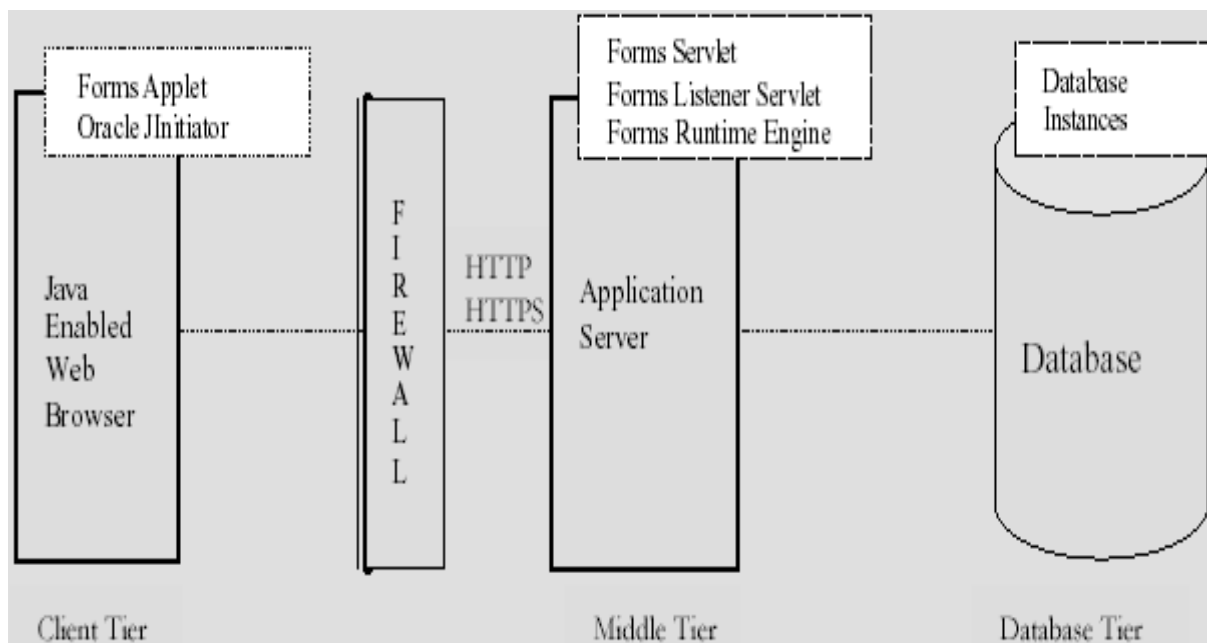
Figura 2.9 – Arquitetura multi-camadas no Oracle Forms Builder, versão 9i (BALA, 2006)

Segundo Gamer (2002), as aplicações do Oracle Forms Builder 9i rodam na Internet devido ao *Oracle Forms Services*, um componente existente no Oracle9i *Application Server*. Ele é composto basicamente por quatro componentes: o *Forms Servlet*, o Java *Client* (ou *Forms Client*), o *Forms Listener Servlet*, e o *Forms Runtime Engine*.

O *Forms Servlet* é um *servlet* Java que cria dinamicamente um arquivo HTML.

O *Forms Client* é um *applet* Java genérico que é dinamicamente baixado e automaticamente carregado na máquina do cliente.

O *Forms Listener Servlet* é responsável pelo gerenciamento da criação dos processos executáveis do Oracle Forms Builder. Ele gerencia também a rede de comunicação entre os clientes e os aplicativos através do servidor da *Web*.

O *Forms Runtime Engine* é responsável por todas as funcionalidades dos formulários, executando o código escrito nas aplicações.

A Figura 2.10 (retirada de GAMER, 2002) demonstra a visão geral da arquitetura do *Forms Services*.
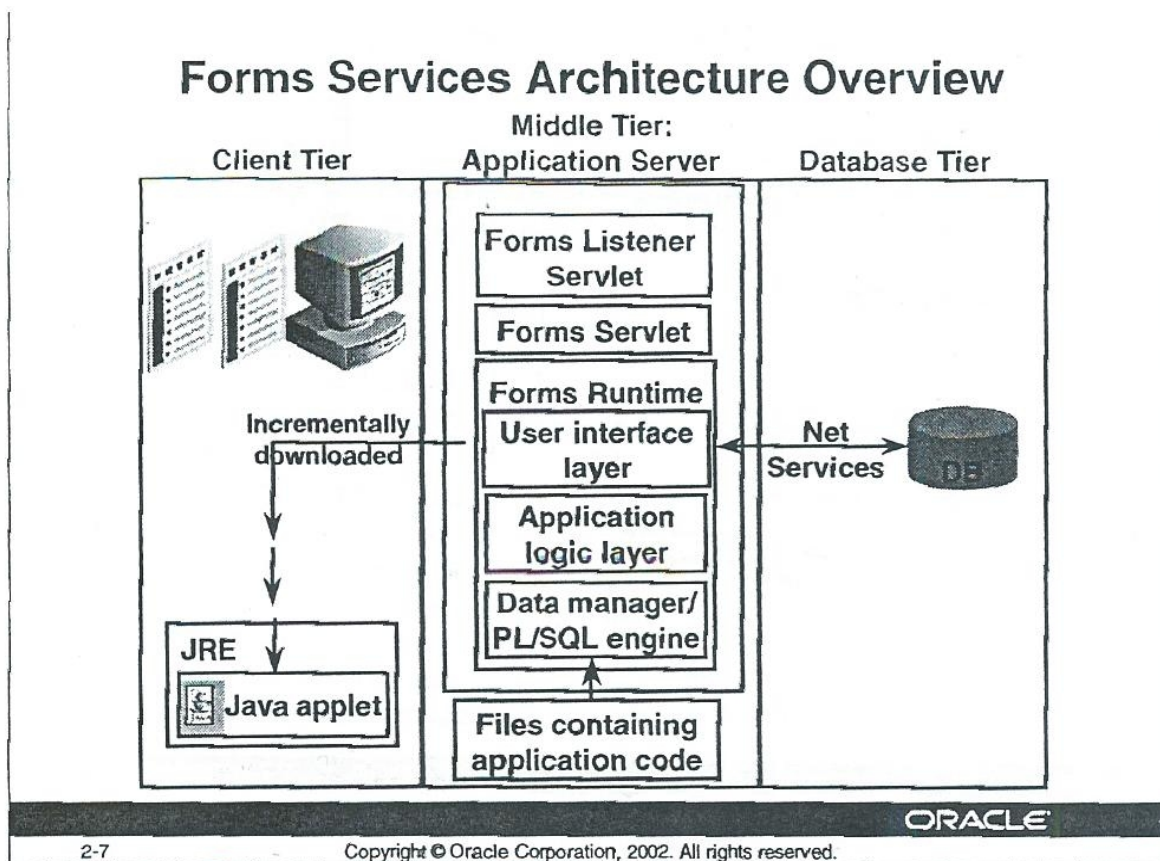
Figura 2.10 – Visão geral da arquitetura do Forms Services (GAMER, 2002)

Embora o Oracle Forms Builder utilize *applets* Java para mostrar os formulários no *browser* do cliente, um desenvolvedor de aplicativos não precisa conhecer Java para programar em *Forms*.

É importante ressaltar que para converter um formulário da versão 6 para a versão 9i basta apenas abrir o formulário no Oracle Forms Builder versão 9i e compilá-lo. Ele será automaticamente atualizado.

3. VISÃO GERAL DO SISTEMA

Neste capítulo é dada uma visão geral de todo o sistema, apresentando os casos de uso e o diagrama entidade-relacionamento. Em seguida, será mostrado o script de geração das tabelas necessárias ao sistema.

3.1. Casos de Uso

Para desenvolver este tutorial foi usado como base um projeto de sistema destinado a uma empresa de sacolas para lojas de grife em geral. Através das especificações das necessidades do usuário, foram desenvolvidos casos de uso com o auxílio da ferramenta Enterprise Architect.

No sistema implementado, o administrador irá controlar o estoque em geral, os cadastros de clientes, os pedidos e o cadastro de vendedores. Na Figura 3.1 é demonstrada a visão geral de como o administrador irá interagir com o sistema.
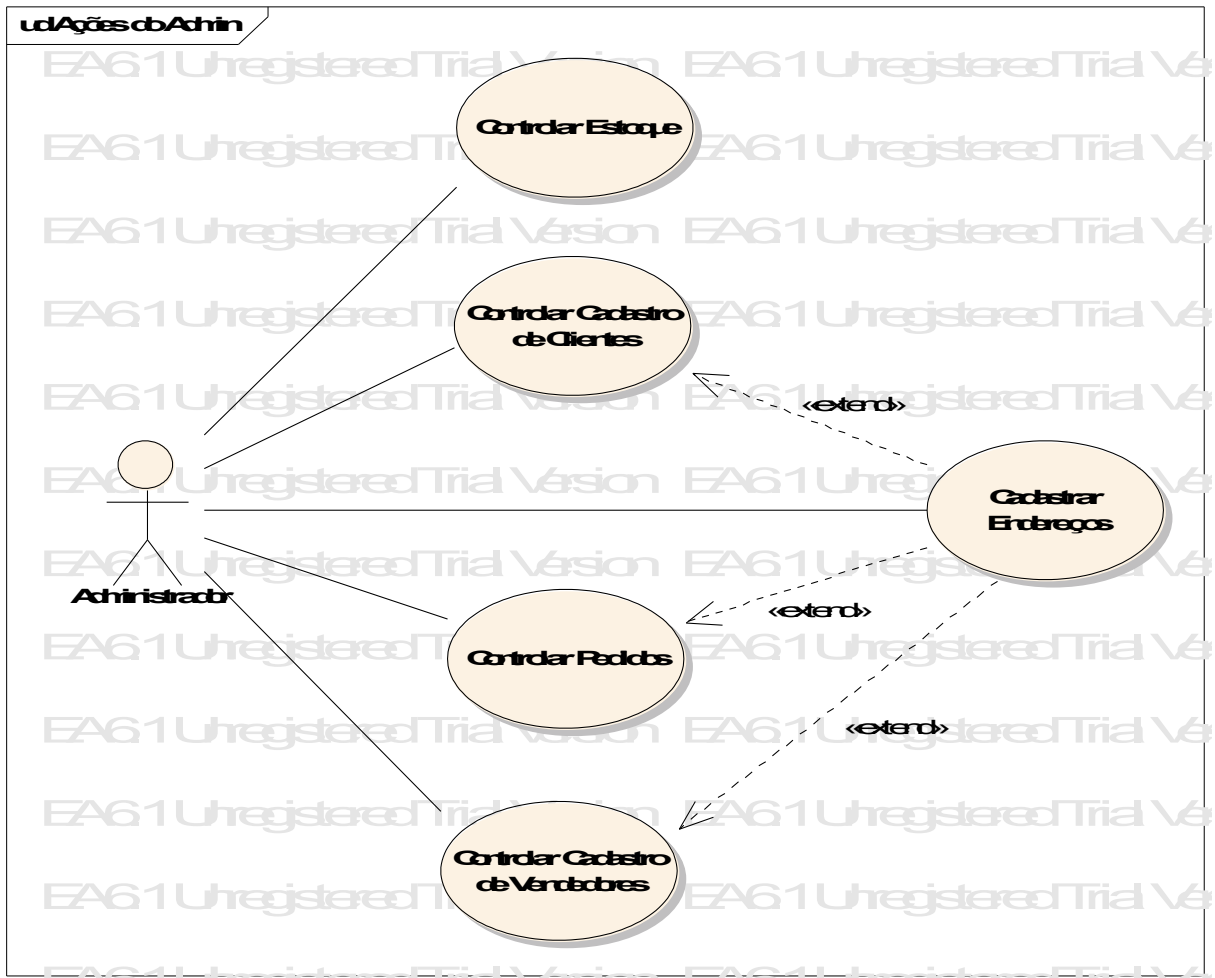
Figura 3.1 – Ações Gerais do Administrador (Casos de Uso)

No módulo de Cadastro de Clientes o administrador poderá cadastrar, alterar, apagar ou consultar clientes, e relacioná-los ao vendedor que o representa. Na Figura 3.2 é exibida a representação deste módulo.

A seguir, os diagramas apresentados para cada funcionalidade utiliza o padrão de análise da UML onde:

- representa uma ação de interface
- representa a fonte de dados
- representa uma ação do projeto
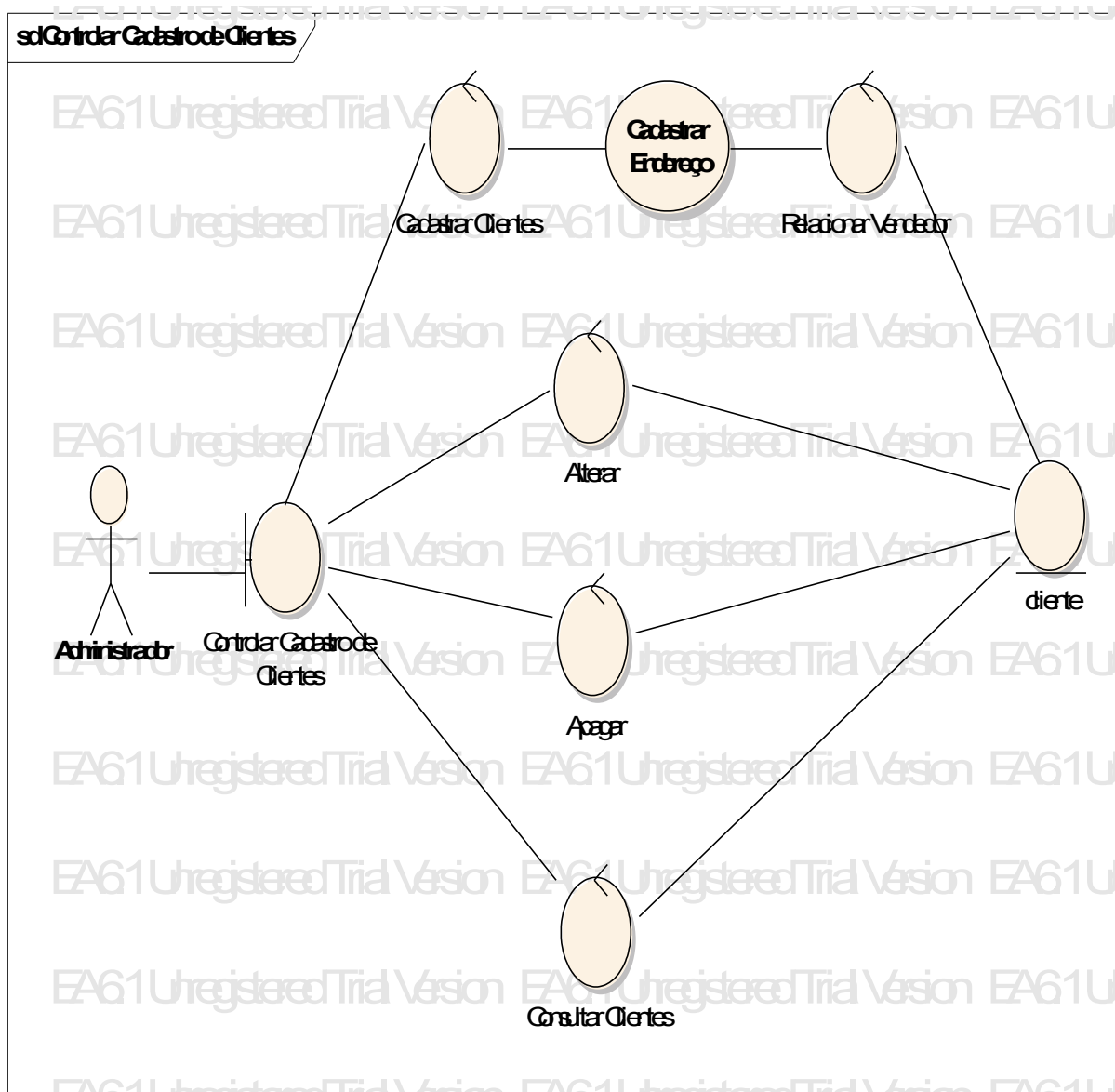
Figura 3.2 – Controle de Cadastro de Clientes

A cada novo cliente, o usuário deverá cadastrar um endereço relacionado a ele. O sistema terá um formulário onde serão cadastrados separadamente os estados e as cidades. Para todos os endereços novos serão associados cidades e estados. A seguir, na Figura 3.3, estão os passos do cadastro de endereços.
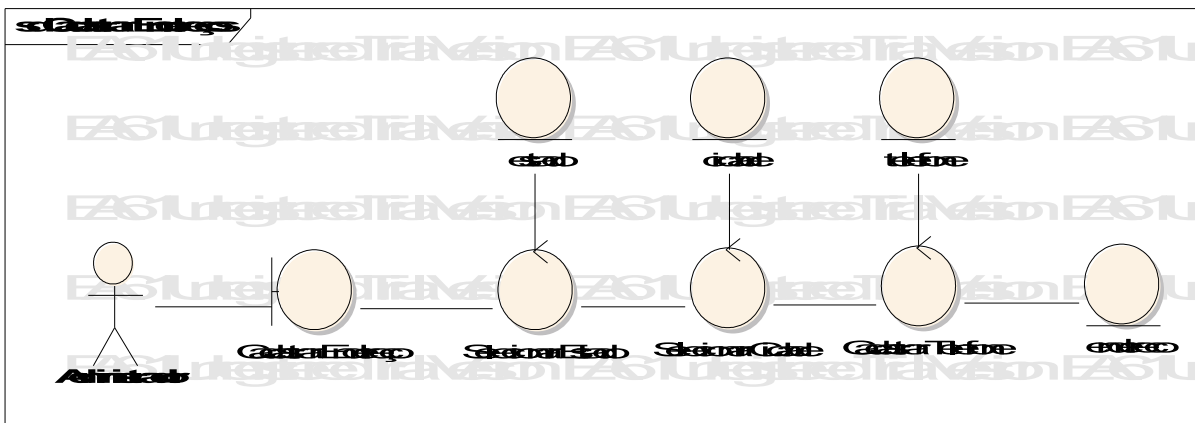
Figura 3.3 – Cadastro de Endereços

Tem-se também um cadastro de vendedores, onde o usuário poderá inserir, consultar, alterar ou apagar registros. Na inserção, o usuário deverá relacionar uma ajuda de custo ao vendedor. Na Figura 3.4 é demonstrada a visão deste módulo.
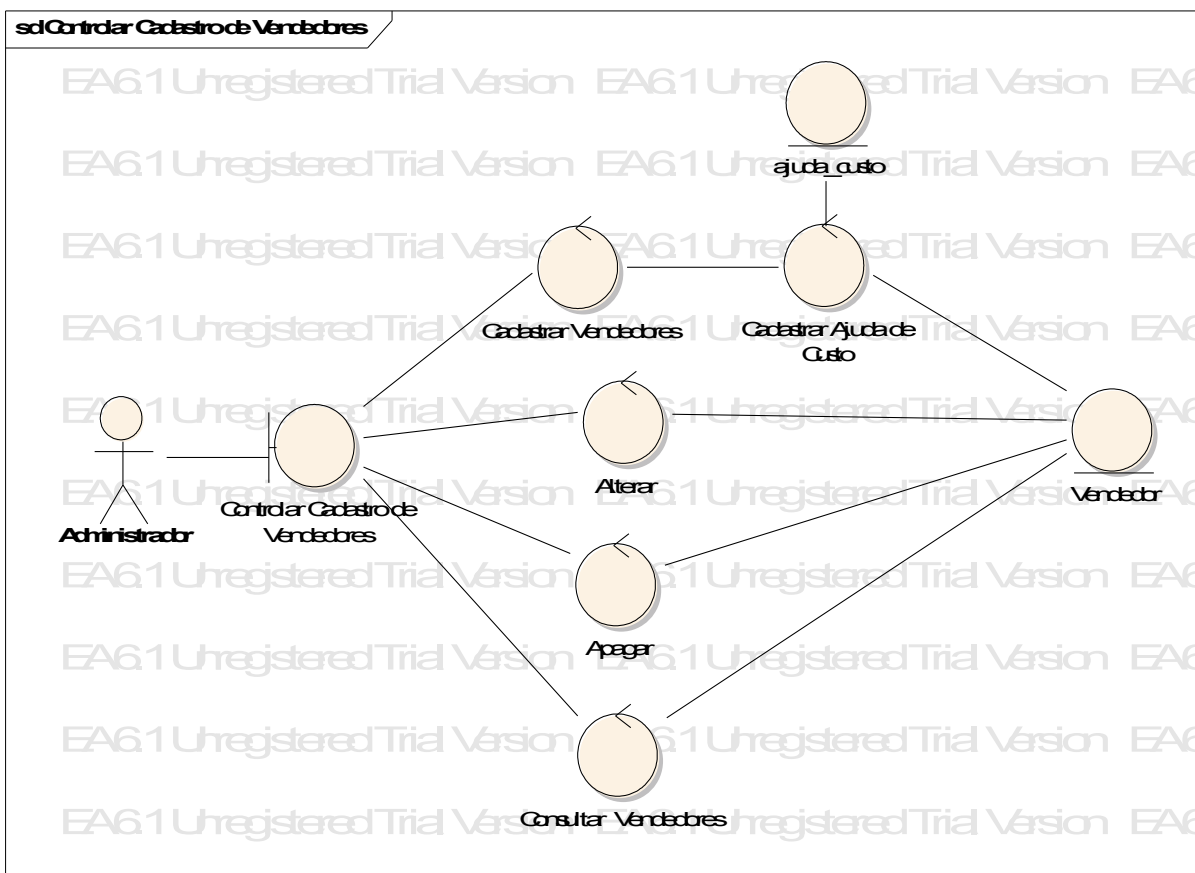
Figura 3.4 – Controle de Cadastro de Vendedores

Na Figura 3.5 é mostrado o módulo de controle de estoque, onde o usuário irá fazer todas as operações de estoque, sendo tanto a entrada quanto a saída de produtos dele. Periodicamente, são feitas contagens do estoque a fim de verificar se o saldo físico confere com o saldo do sistema.
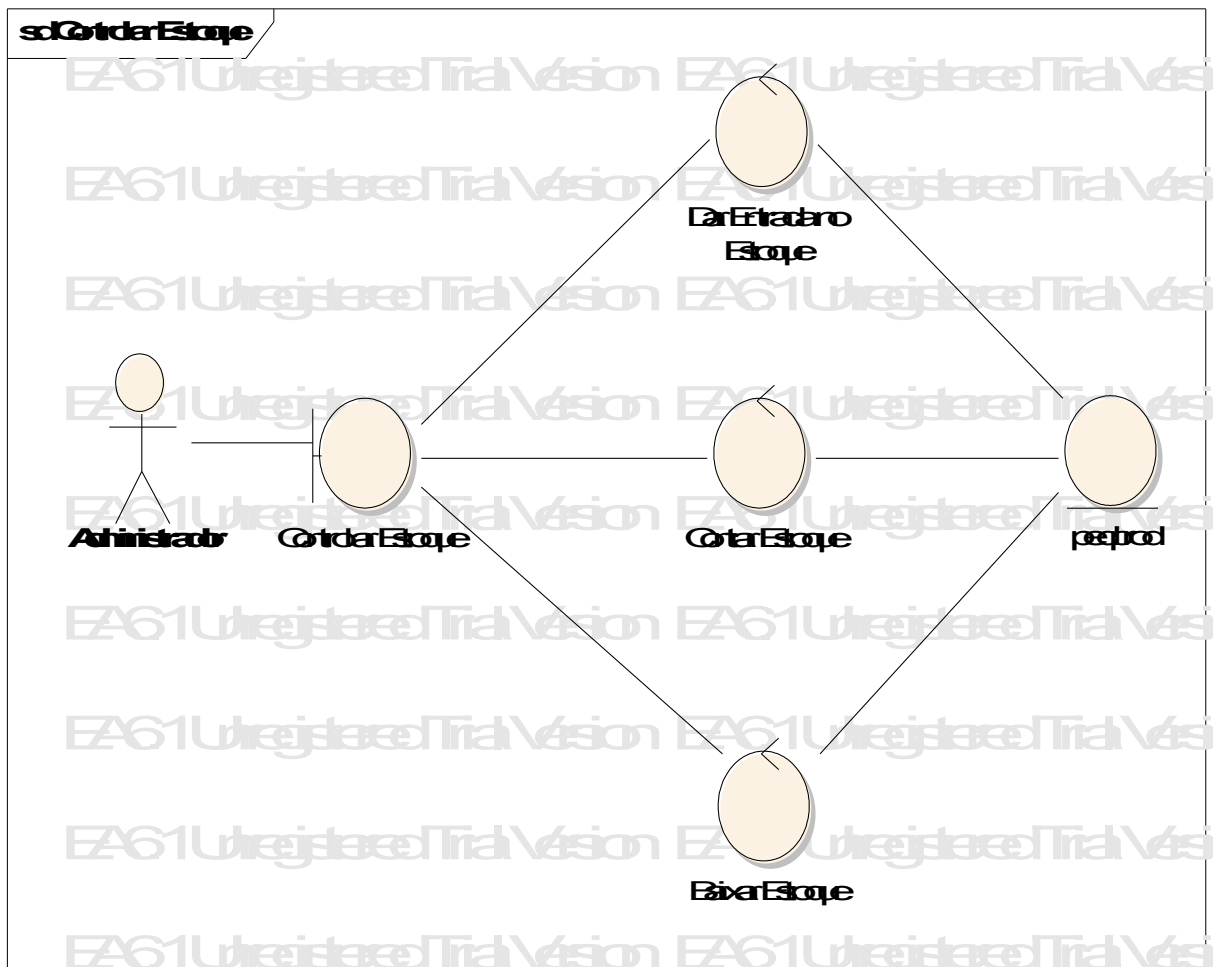
Figura 3.5 – Controle de Estoque

Na Figura 3.6 é exibido o módulo onde serão controlados todos os pedidos da empresa.
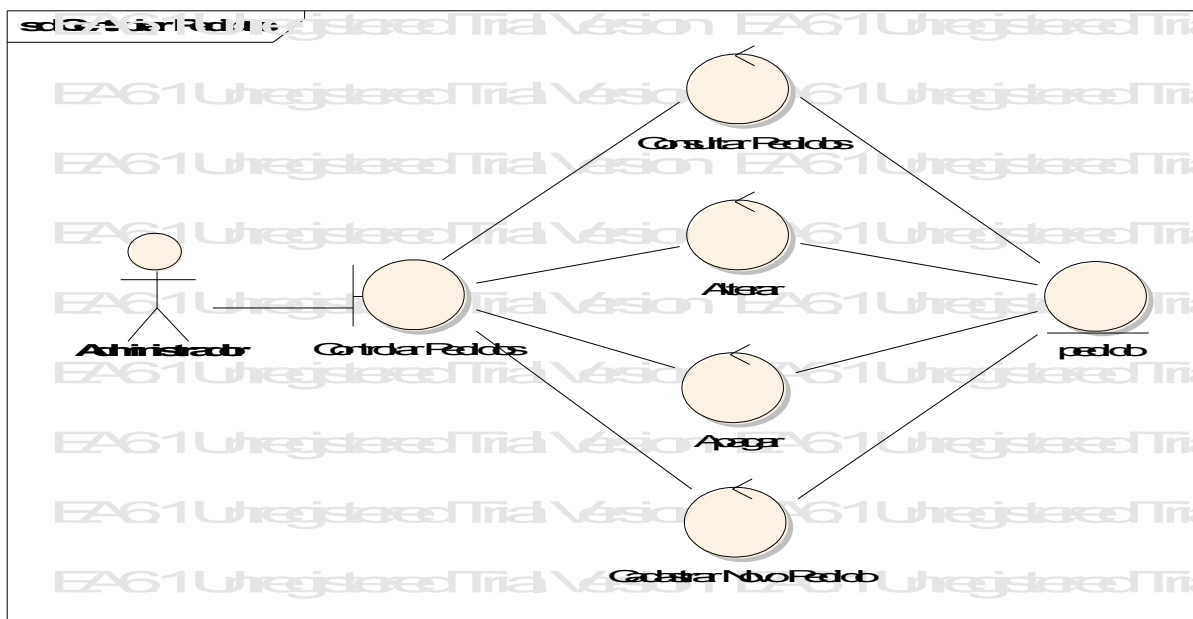
Figura 3.6 – Controle de Pedidos

No controle de produtos o usuário irá cadastrar todos os tipos de sacolas trabalhadas pela empresa, detalhando desde as medidas até a densidade do produto. Nesse módulo também será possível consultar, alterar e apagar registros. A Figura 3.7 representa este módulo.
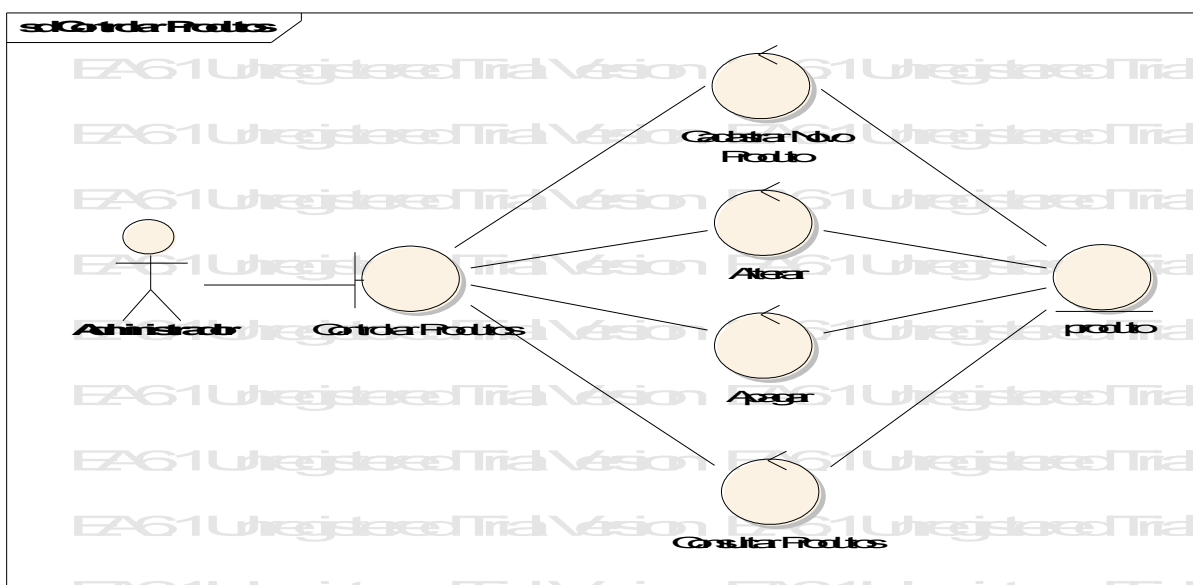
Figura 3.7 – Controle de Produtos

3.2. Diagrama Entidade-Relacionamento

A seguir, na Figura 3.8, é exibido o diagrama entidade-relacionamento gerado para o desenvolvimento do sistema exemplo. O script de geração das tabelas que serão utilizadas está no Anexo A.
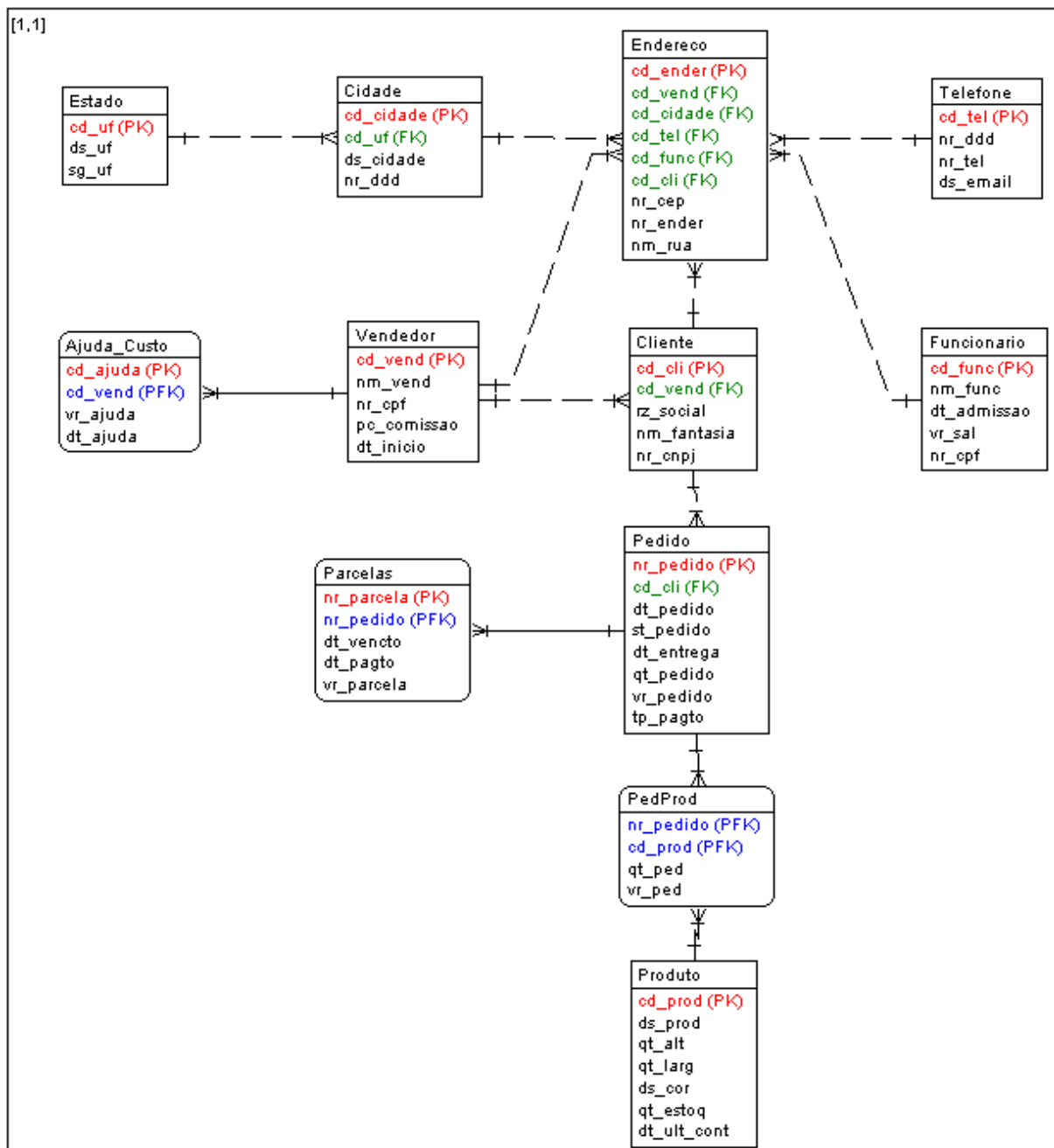
Figura 3.8 – Diagrama Entidade-Relacionamento

4. TUTORIAL

Será iniciado agora o tutorial da Ferramenta Oracle Forms, onde será construído o sistema mencionado no capítulo anterior.

4.1. Iniciando o Oracle Forms

Ao iniciar o Oracle Forms a tela da Figura 4.1 será visualizada.
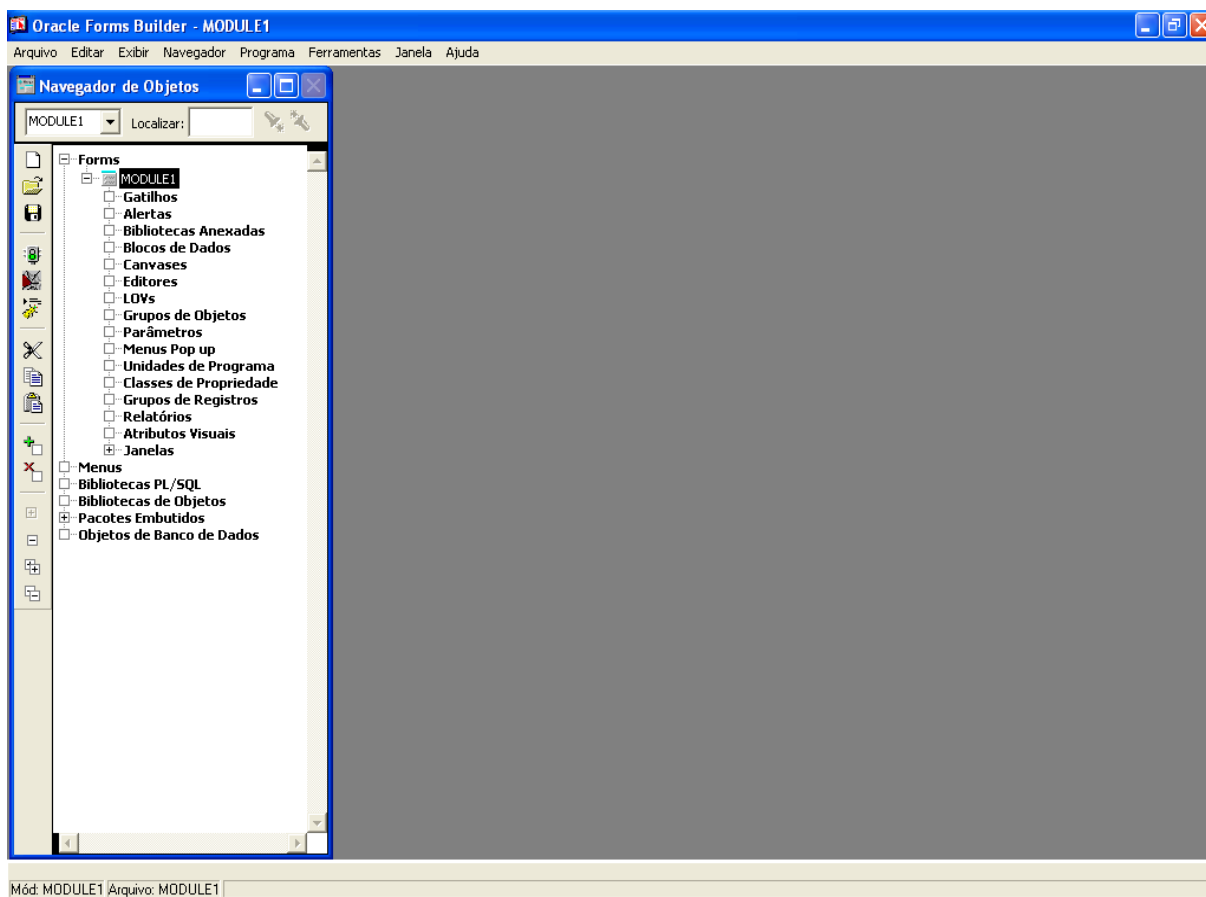
Figura 4.1 – Tela Inicial do Oracle Forms

Para visualizar a Paleta de Propriedades basta clicar na Tecla F4, e para visualizar o Canvas tem-se o atalho com a Tecla F2. Outro atalho importante é a Tecla F3, que traz o

Navegador de Objetos. É importante ter em mente esses atalhos porque ao longo do desenvolvimento dos aplicativos necessita-se constantemente da mudança rápida de uma janela para outra.

4.2. Criando um Bloco de Dados Baseado em Tabela

Inicialmente, será criado um bloco para o cadastro de estados. Para criá-lo, vá ao "Navegador de Objetos" e clique em "Bloco de Dados". Então vá à barra de ferramentas que se encontra na lateral e clique em (Criar). Será então aberta a tela da Figura 4.2, em que o usuário escolhe se irá fazer um bloco de dados usando o assistente ou fazendo-o manualmente. Geralmente opta-se por desenvolver um bloco manualmente quando se deseja apenas um bloco de controle, sem que ele seja baseado em qualquer tabela:
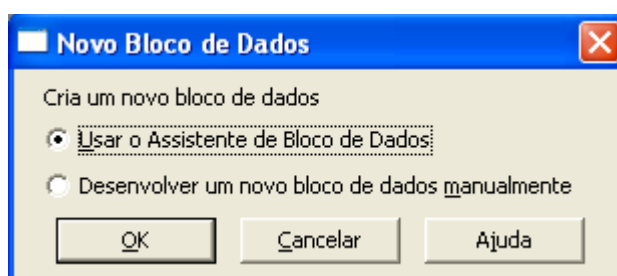
Figura 4.2 – Escolha do Tipo de Bloco de Dados

Selecione a opção de usar o assistente e ao clicar em Ok será mostrada uma tela onde será escolhido se o bloco será baseado em uma tabela ou visão, ou se será baseado em um procedimento armazenado. Este último é mais usado por programadores avançados, por isso, serão usadas tabelas. Ao clicar em Avançar, deverá ser escolhida qual tabela utilizar no bloco. Neste caso, a tabela Estado. Em seguida serão mostradas todas as colunas da tabela para que

sejam indicadas quais serão utilizadas pelo aplicativo. Selecione as três colunas para o aplicativo.

Clicando em Avançar, o Forms dá a opção de relacionar o bloco que está sendo criado com um já existente. Não será utilizada esta opção agora. Clique em Avançar novamente e clique em Encerrar. Automaticamente será aberto o Assistente de Layout, onde se configura a aparência do aplicativo.

Primeiramente irá aparecer uma lista de seleção única onde deve-se escolher em qual Canvas a aplicação ficará. Como ainda não tem nenhum, opte pela opção Novo Canvas e deixe o tipo dele como Conteúdo. Clicando em Avançar escolha quais os campos que serão mostrados no Canvas, já que nem todos os campos precisam estar visíveis na aplicação. Para esse aplicativo serão necessários os três. Clique em Avançar novamente e aparecerá a opção de ajustar os tamanhos dos campos, mas não é necessário fazer isso agora, pois através do Visualizador de Canvas essa tarefa fica mais fácil.

Clique em Avançar e escolha que tipo de Layout será utilizado na aplicação: Formulário, que mostra um registro de cada vez; ou Tabular, que mostra vários registros. Selecione a opção Tabular e clique em Avançar. Na nova tela tem-se que dar um nome ao nosso Canvas, portanto digite Cadastro de Estados no campo Título do Quadro. No campo Registros Exibidos opte por mostrar 10 registros e ligue o botão de Marcação de Item para que seja mostrada a barra de rolagem. Clique em Avançar e pronto, o primeiro bloco do Oracle Forms foi criado. Na Figura 4.3 é mostrada a aplicação que foi construída.
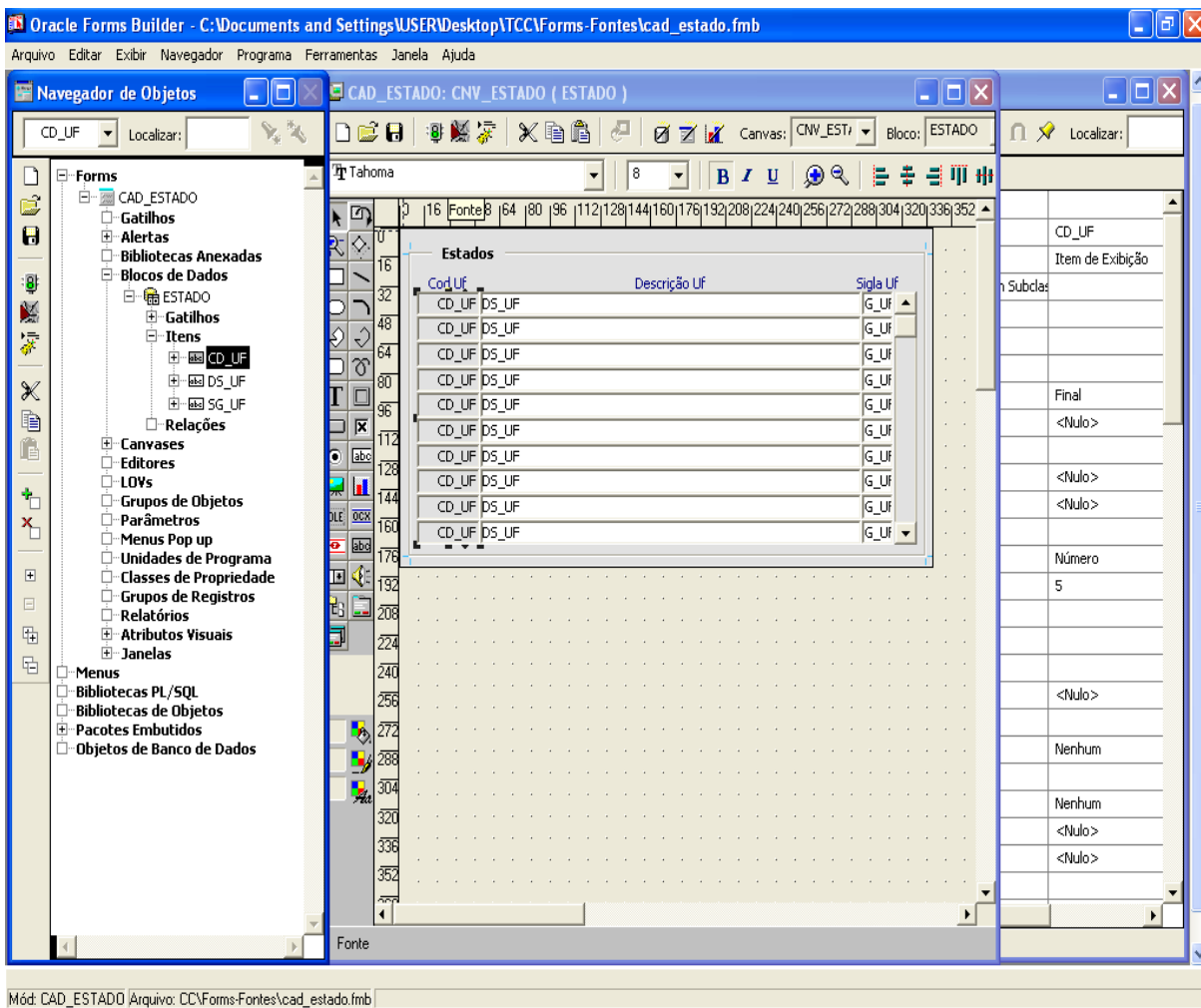
Figura 4.3 – Aplicação de Cadastro de Estados

Agora selecione a coluna CD_UF no Navegador de Objetos e vá à Paleta de Propriedades. Mude o tipo de item para Item de Exibição. Com isso o usuário não poderá entrar com nenhum valor nessa coluna, pois para evitar erros humanos no cadastro será criada uma trigger que, ao ser inserido um novo registro, preenche esta coluna automaticamente.

Crie agora a trigger. Vá ao bloco Estado, clique em Gatilhos e em seguida em Criar. Então a tela da Figura 4.4 será mostrada.
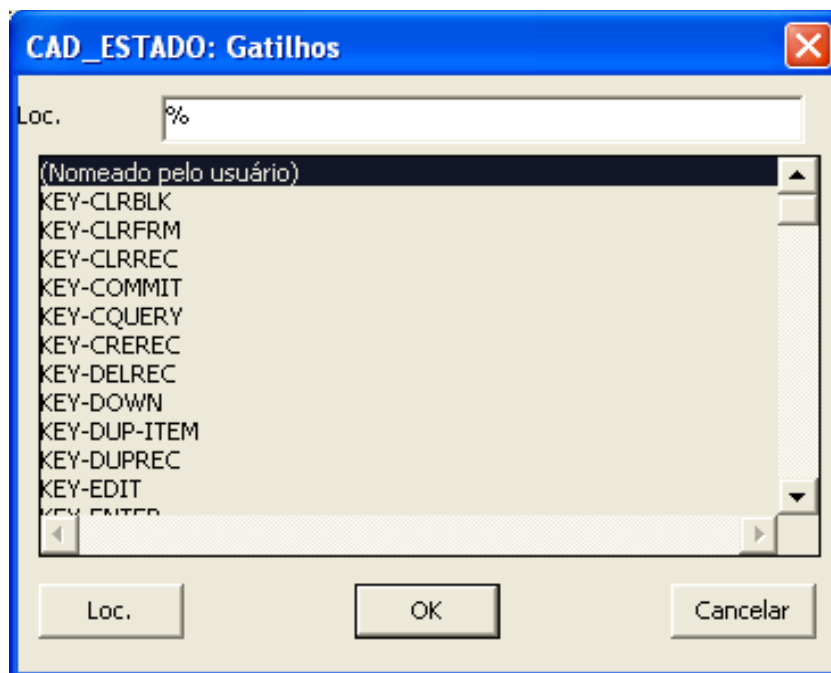
Figura 4.4 – Tela de Escolha dos Gatilhos

Escolha então a trigger PRE-INSERT e dentro do editor de texto digite o bloco PL/SQL da trigger:

```
begin
      select nvl(max(cd_uf),0)+1
        into :estado.cd_uf
       from estado;
end;
```

Agora compile o aplicativo e execute-o. Faça inserções de alguns estados para que possam ser usadas mais tarde.

4.3. Criação de uma LOV (Lista de Valores)

Para essa Seção é necessário o bloco de Cadastro de Cidades. Para fazê-lo siga os mesmos passo do aplicativo de Cadastro de Estados, criando um novo formulário e um bloco baseado na tabela de Cidades.

A diferença entre os dois aplicativos é que será preciso cadastrar o código no estado referente à cidade. Para isso será feito o uso de uma LOV. Também é preciso criar um campo não baseado em tabela, que é um campo apenas para mostrar algum dado que não está na tabela do bloco criado.

Como a tabela Cidades armazena o código do estado, e para o usuário essa informação não é completa, tem-se que criar uma trigger POST-QUERY a nível de bloco para que ela alimente o campo que irá mostrar a sigla do estado relacionado à cidade. Vá então em Gatilhos, logo abaixo do nome do bloco e crie uma trigger com a seguinte instrução SQL:

```
begin
  select sg_uf
    into :cidade.sg_uf
   from estado
  where cd_uf = :cidade.cd_uf;
end;
```

Como não é preciso mostrar o campo CD_UF vá à Paleta de Propriedades e altere o campo Canvas para nulo.

Após criar o campo SG_UF, vá à Paleta de Propriedades e altere o campo Item do Banco de Dados para Não, e altere o campo Canvas escolhendo o nome que você deu a ele na criação do bloco. Arrume então a posição dos campos para que fiquem como na Figura 4.5. Não se esqueça de criar uma trigger para preencher o campo CD_CIDADE automaticamente.
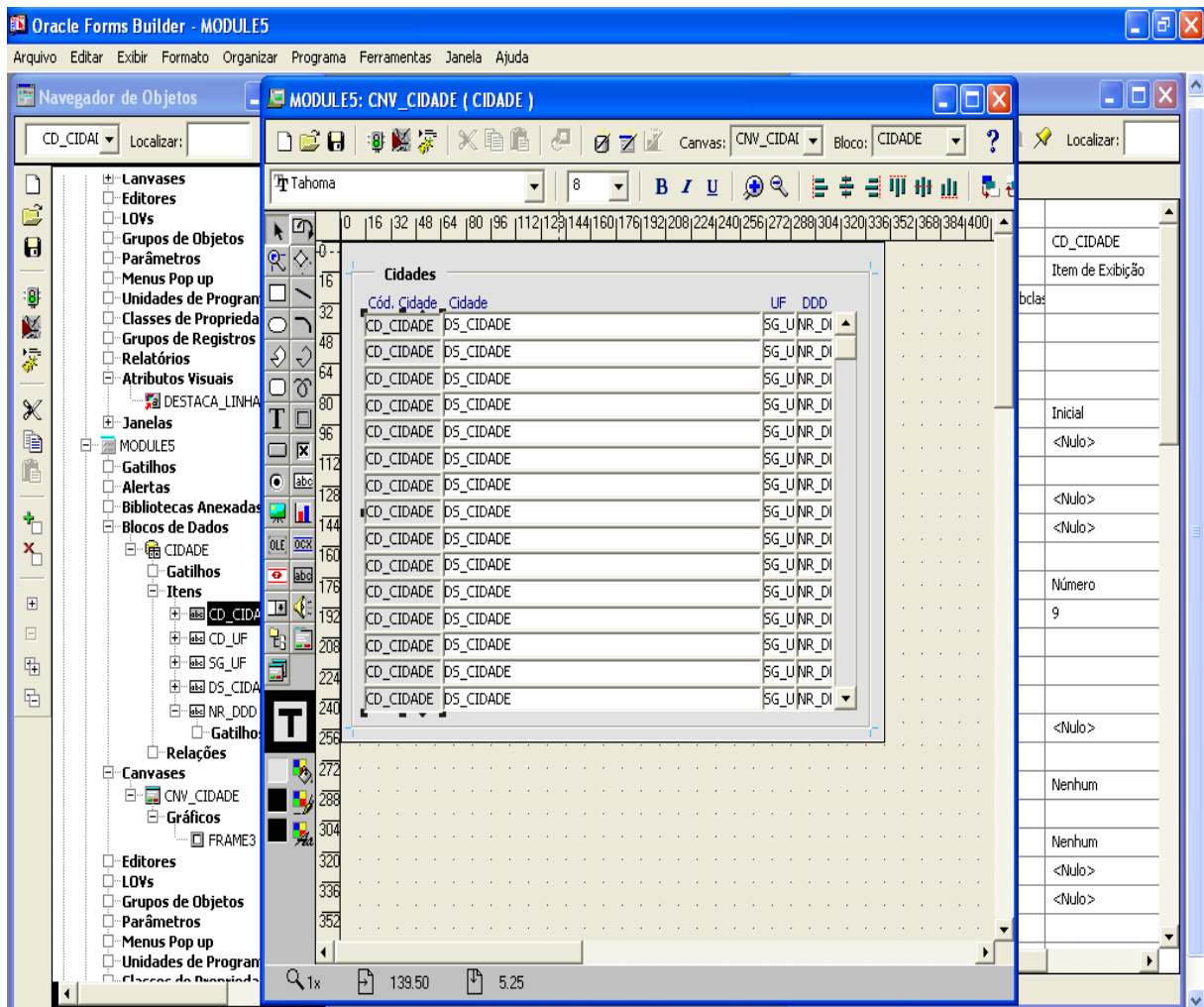
Figura 4.5 – Tela do Aplicativo Cadastro de Cidades

Para criar a LOV vá ao Navegador de Objetos, clique em LOVs e depois em Criar. Opte por usar o assistente e clique em Ok. No item de texto Instrução de Consulta SQL entre com a instrução que trará as colunas CD_UF e SG_UF:

```
select cd_uf, sg_uf
  from estado
```

Clique em Avançar e selecione as duas colunas para a LOV. Clique em Avançar e no campo Valor de Retorno coloque as duas colunas referentes ao campo do bloco, CIDADE.CD_UF e CIDADE.SG_UF. Clique em Avançar novamente e dê um título à LOV. Depois escolha quantas linhas serão mostradas e logo após selecione quais serão os campos

que receberão os dados da LOV. Selecione os dois campos e clique em Avançar, e a LOV está criada.

Para consultar no aplicativo a LOV que foi criada, deixe o cursor sobre o campo SG_UF e vá à barra de menu, em Editar, e clique em Exibir Lista. É só selecionar o registro desejado. A consulta da LOV está na FIGURA 4.6.
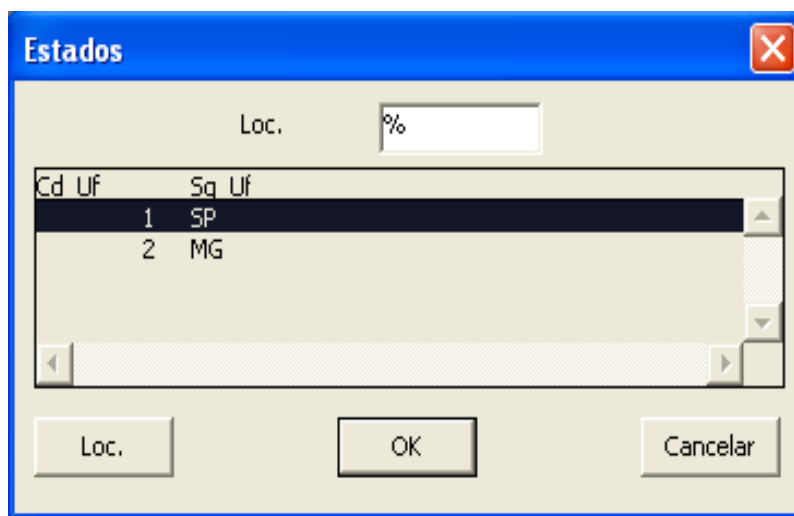
Figura 4.6 – Consulta de uma LOV

Para os aplicativos de endereços, clientes, funcionários e vendedores, as funcionalidades usadas são praticamente as mesmas vistas até aqui, com triggers de validação e LOVs para auxiliar nas inserções.

4.4. Criação de um Menu de Alerta

Freqüentemente necessita-se de um menu de alerta para mostrar ao usuário opções de caminhos a seguir, ou apenas para trazer mensagens de erro. Nesta Seção é criado um menu de alerta genérico para mostrar ao usuário quando algum erro ocorrer.

Vá à opção Alertas, no Navegador de Objetos e clique em Criar. Vá à Paleta de Propriedades e mude o nome para ALERTA_GENERICO. No item Etiqueta do Botão 2 apague o que está escrito dentro. O alerta já está criado.

Agora, precisa-se criar um procedimento que altere o conteúdo da mensagem do menu de alerta em tempo de execução. Vá até Unidades de Programa no Navegador de Objetos e clique em Criar. Então crie o seguinte procedimento:

```
PROCEDURE alerta1
(
    msg                     varchar2      -- parâmetro de entrada
)
IS
  al_id       alert;           -- variável que irá receber o ID do alerta
  al_button   number;               -- variãvel que irá receber o alerta
BEGIN
  al_id := Find_Alert('ALERTA_GENERICO');
   Set_Alert_Property(al_id, alert_message_text, msg );  -- muda o texto do
alerta
  al_button := Show_Alert( al_id );
END;
```

Agora sempre que for preciso mostrar uma mensagem de aviso será usado este Alerta. Como exemplo é criada a trigger de validação do campo SG_UF.

No bloco Estado, caso o usuário entre com um valor diretamente no campo SG_UF, coluna essa que não pertence à tabela Cidades, precisa-se de uma trigger que coloca o código referente àquela sigla no campo CD_UF. Para isso é criada a trigger WHEN-VALIDATE-ITEM no item SG_UF e colocado a seguinte instrução nela:

```
begin
  select cd_uf
    into :cidade.cd_uf
   from estado
  where sg_uf = :cidade.sg_uf;
exception
   when no_data_found then
        -- chamar a procedure que altera a mensagem de alerta
        alerta1('Sigla de estado não cadastrada na tabela ESTADO');
        -- cancela o restante da trigger
        raise form_trigger_failure;
   when too_many_rows then
        alerta1('Sigla de estado cadastrada mais de uma vez na tabela ESTADO');
        raise form_trigger_failure;
end;
```

Quando for tentado inserir uma sigla de estado que não está cadastrada o Oracle Forms nos trará o menu de alerta da Figura 4.7.
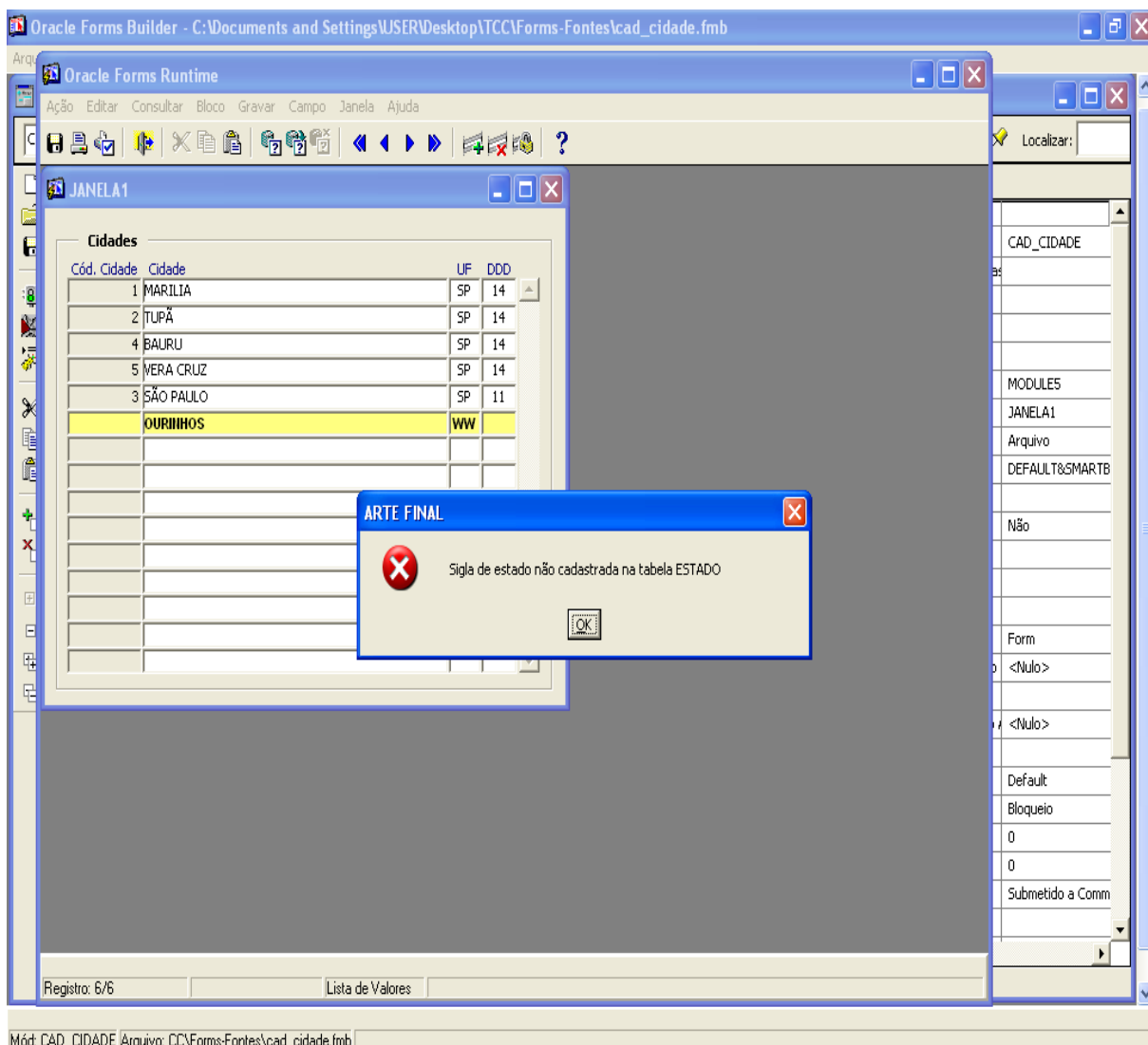
Figura 4.7 – Alerta de Erro

4.5. Criação de um Bloco Mestre-Detalhe

Para a criação do aplicativo de pedidos será necessária a criação de um bloco mestre e dois blocos detalhe. O bloco mestre será baseado na tabela Pedido, e os blocos detalhe nas tabelas Parcelas e PedProd.

O primeiro passo será criar o bloco mestre. Crie-o com o auxílio do Assistente de Bloco de Dados como mostrado no Seção 4.2, e crie uma LOV para que seja relacionado ao

cliente que fez o pedido. Coloque também alguns itens não baseados em tabela para mostrar o endereço do cliente. Estes itens devem ser populados em triggers que sejam disparadas tanto na inserção quanto na consulta. Terminada a criação deste bloco dever-se ter algo parecido à Figura 4.8.

Figura 4.8 – Bloco Mestre Baseado na Tabela Pedido

Terminada a criação do bloco mestre crie o primeiro bloco detalhe. Ele será baseado na tabela Parcelas. Para criá-lo, use o Assistente de Bloco de Dados. Opte por fazer o bloco baseado em tabela e selecione a tabela Pedido. Clique em avançar e será apresentada a Figura 4.9. Desmarque a opção “Fazer junção automática dos blocos de dados” e clique em Criar Relacionamento.

Figura 4.9 – Criando Relacionamento entre Blocos

Em seguida escolha por criar a relação baseada em uma condição de união. Escolha a tabela que irá ser relacionada com a que está sendo criada. Agora indique quais colunas vão igualar as tabelas. Faça como na Figura 4.10 e escolha a coluna nr_pedido.
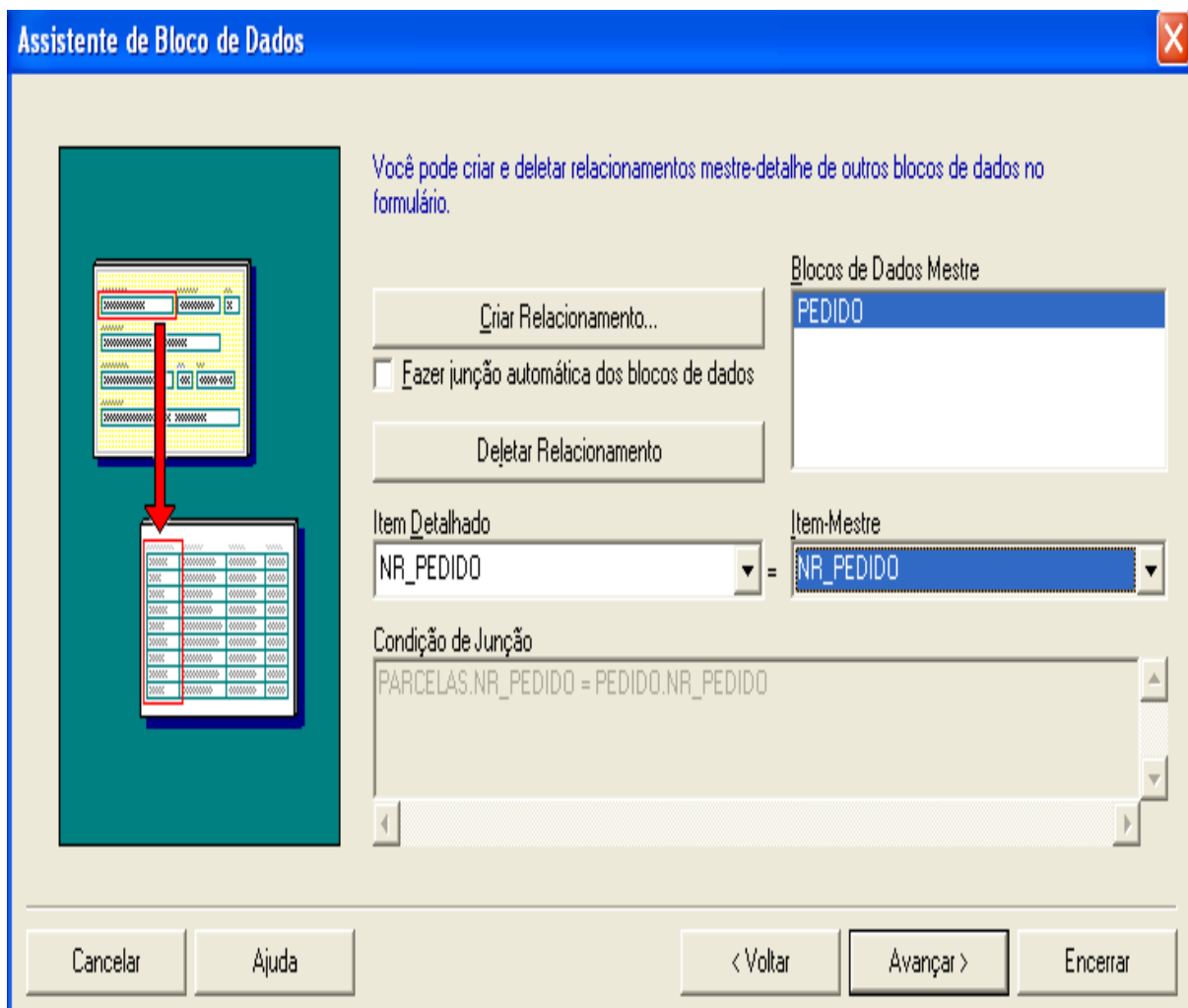
Figura 4.10 – Escolha das Colunas de União do Relacionamento

Agora, com o relacionamento criado, o processo de criação do bloco é igual ao de outro bloco qualquer. O Canvas ficará com a aparência da Figura 4.11.
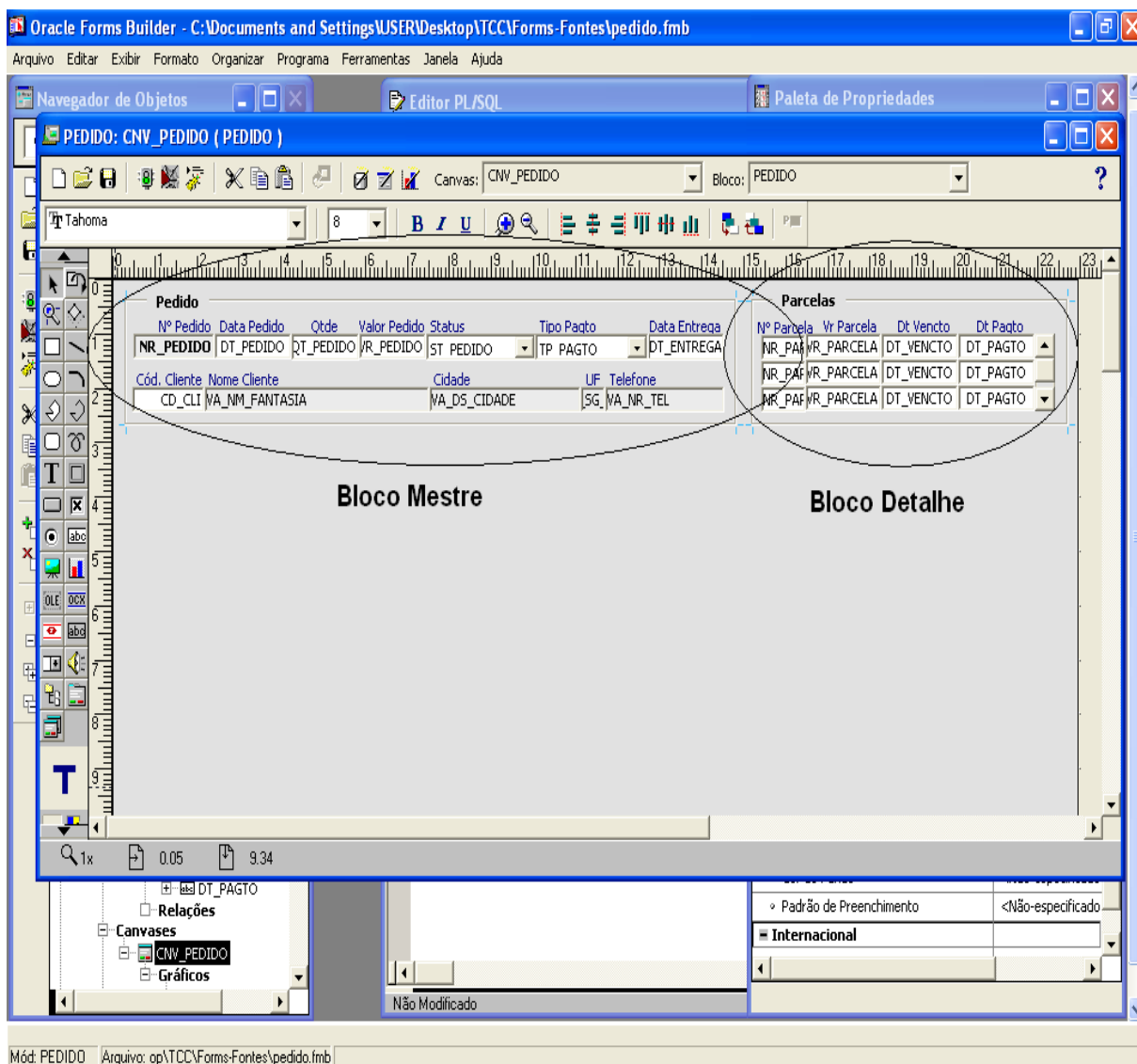
Figura 4.11 – Aplicativo com Blocos Relacionados (Mestre-Detalhe)

Para criar o bloco detalhe relacionado à tabela PedProd deve-se seguir os mesmos passos da tabela Parcelas. Posicione-a logo abaixo ao bloco de pedido. Então, ao finalizar este aplicativo tem-se a janela apresentada na Figura 4.12.

Figura 4.12 – Aplicativo de Cadastro de Pedidos

4.6. Criação de um Bloco de Controle

Devido ao fato de ser necessário chamar apenas seis aplicativos, ao invés de um menu, crie um bloco de controle com botões que chamam o respectivo aplicativo. Quando tem-se um sistema com muitos aplicativos, o recomendado é a criação de menu.

Para criar o bloco de controle, vá ao navegador de objetos e selecione Bloco de Dados. Em seguida clique em criar e opte por desenvolver um bloco manualmente, como apresentado na Figura 4.13.
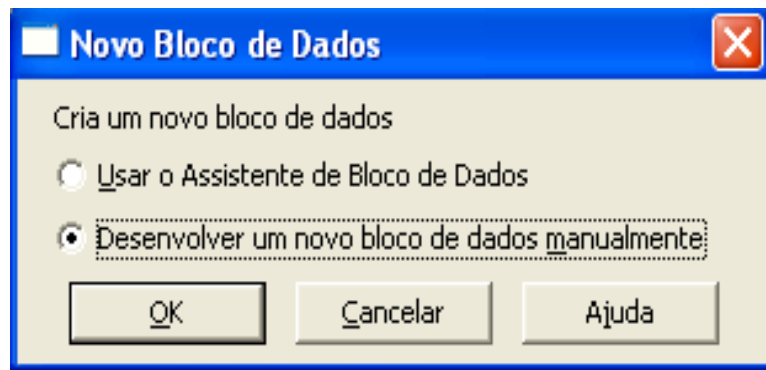
Figura 4.13 – Escolha do tipo de bloco de dados

Selecionado o tipo de bloco de dados, crie agora todos os botões que chamarão as aplicações. Para isso, crie um item, e na Paleta de Propriedades mude o Tipo de Item para tecla. Em seguida selecione o Canvas onde o botão vai aparecer e no item Etiqueta coloque qual será a etiqueta do botão.

Com o botão criado, agora faça a trigger que será disparada quando o botão for pressionado. Essa trigger irá chamar o respectivo aplicativo relacionado. Crie a trigger a nível de item WHEN-BUTTON-PRESSED e insira o seguinte código PL/SQL:

```
begin
  call_form('cad_estado',no_hide,do_replace,no_query_only,no_share_library_data);
end;
```

Para chamar todos os outros aplicativos, crie botões seguindo este mesmo modelo apresentado anteriormente. O menu ficará igual à Figura 4.14.
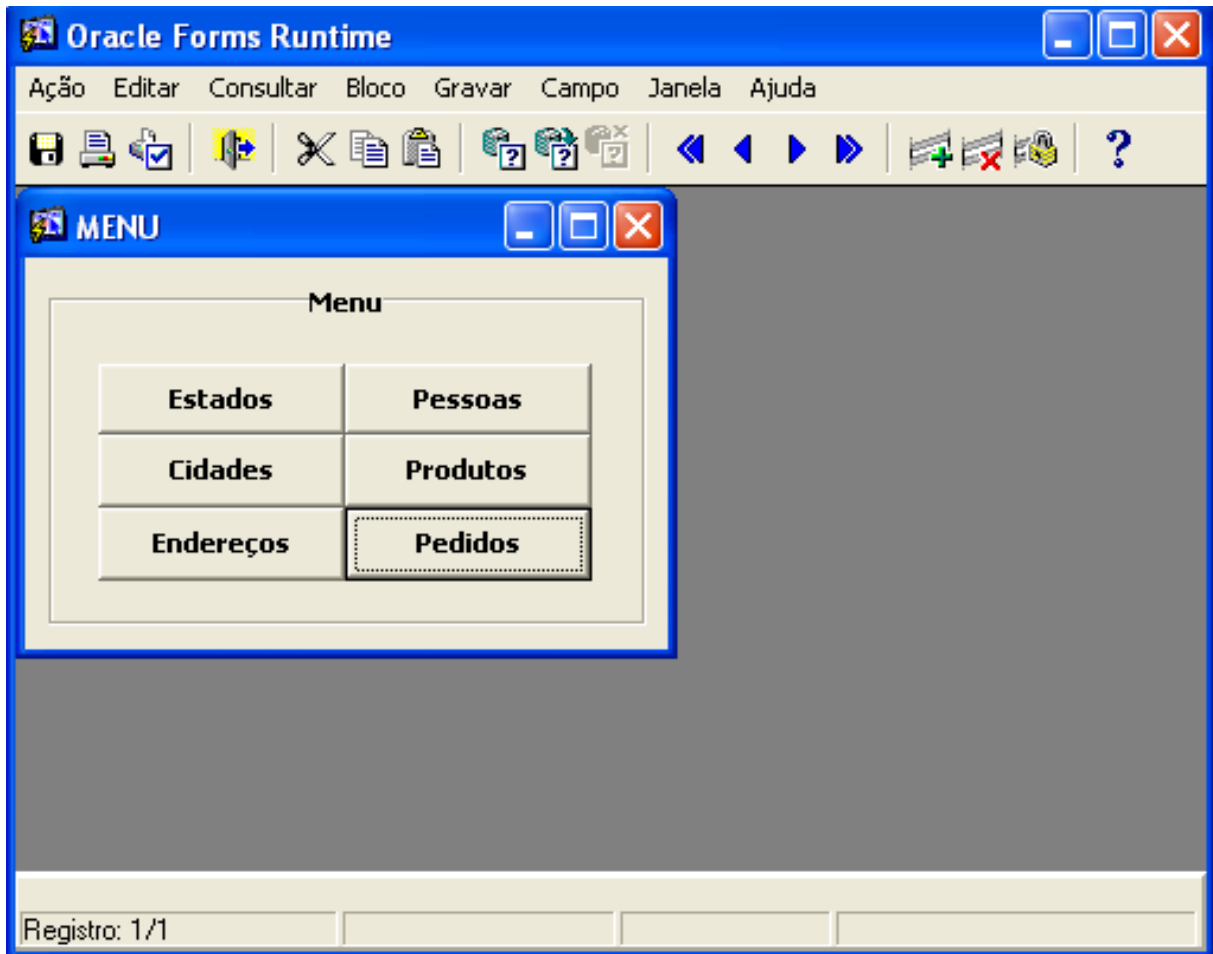
Figura 4.14 – Aplicativo de Menu baseado em um Bloco de Controle

CONCLUSÕES

Como pode se observar neste tutorial, o Oracle Forms Builder é uma ferramenta de fácil entendimento, e de fácil programação. Devido ao fato de a ferramenta fazer a comunicação com o banco de dados sem que haja a necessidade de códigos extensos, o programador pode se preocupar apenas com a interface a ser mostrada ao usuário e onde serão armazenadas as informações.

Foi possível observar também que algumas operações do Oracle Form Builder são necessárias em qualquer aplicação a ser desenvolvida. A exemplo disso têm-se os alertas que são dados em qualquer mensagem de erro, as triggers que são disparadas a cada evento, e os blocos baseados em tabela e blocos relacionados.

Ao final da implementação do sistema nota-se que o caso de uso utilizado como exemplo para o tutorial, pode ser usado também para outros tipos de empresa de porte pequeno, necessitando apenas de algumas modificações particulares.

O trabalho atual restringe-se aos conceitos e técnicas para a construção de formulários onde serão feitos consultas e cadastros. Para a geração de relatórios é preciso a utilização de outra ferramenta da Oracle, o Oracle Reports Builder. Através dele construímos relatórios trazendo quais e como os dados de determinadas tabelas serão impressos. Um trabalho futuro será a criação de um tutorial desta ferramenta.

Este trabalho atendeu seu objetivo em dispor aos leitores desta monografia as principais características de uso da Ferramenta Developer Kit da Oracle® no desenvolvimento de Aplicativos de Banco de Dados Relacional mostrando e explicando as duas últimas versões existentes, a 6 e a 9i.

REFERÊNCIAS BIBLIOGRÁFICAS

ABBEY, M.; COREY, M. J. **Oracle: Guia do Usuári***o*. São Paulo: Makron Books, 1997.

BALA, Kiran. **Oracle 9i Forms Listener Servlet**. Disponível em: <http://oracle.ittoolbox.com/documents/peer-publishing/oracle-9i-forms-listener-servlet-2361>. Acesso em : 22 nov. 2006.

FERNANDES, L. **Oracle 9i: Para Desenvolvedores Oracle 6i – Curso Completo**. Axcel Books, 2002.

FRAINER, A. S. **Planos na Interação Homem-máquina.** 1990. 60f. Dissertação (Mestrado em Ciência da Computação) - Universidade do Rio Grande do Sul, Porto Alegre, 1990.

GAMER, P. **Oracle 9i Forms Developer: New Features.** Oracle Corporation, 2002.

LOH, Y. & VAVASSORI, F. B. **Método Heurístico para Avaliação e Projeto de Interfaces Homem-Software.** Disponível em: <http://atlas.ucpel.tche.br/~loh/interfac.htm>. Acesso em : 18 nov. 2005.

LUCENA, F. N. & LIESENBERG, H. K. E. **Interfaces Homem- Computador: Uma Primeira Introdução.** Disponível em: <http://www.dcc.unicamp.br/proj-xchart/start/indice .html>. Acesso em : 22 nov. 2005.

LUCENA, M. **Análise dos recursos para construção de interfaces e da fundamentação do projeto de interfaces.** Disponível em: <http://csg.uwaterloo.ca/~marisa/publicat/exame/exame.html>. Acesso em : 03 março 2006.

MORELLI, E. T. **Oracle 9i Fundamental: SQL, PL/SQL e Administraçã***o*. São Paulo: Érica, 2002.

OLIVEIRA, R. T. **Guia de Consulta Rápida Oracle 10g SQL**. São Paulo: Novatec, 2004.

PATABALLA, N.; NATHAN P. **Introduction to Oracle 9i: PL/SQL. Student Guide**. Volume 1. Junho de 2001.

PODA, A. **Guia de Consulta Rápida Oracle Forms 6i**. São Paulo: Novatec, 2001.

RAMALHO, J. A. A. **Oracle – Personal Oracle 7.3 & Power Objects** *2.0*. São Paulo: Makron Books, 1997.

RODRIGUES, Anderson H. **Sistemas Multicamadas com Delphi DataSnap e DBExpress**: Conceitos, Implementação e Macetes. Florianópolis: VisualBooks, 2002.

SHNEIDERMAN, B. Direct Manipulation: A step Beyond Programming Languages**, IEEE Computer,** Vol. 16, No. 8, p. 57-69. Aug 1983.

SILVA, J. C. A.. **Uso de gabaritos configuráveis para desenvolvimento de interfaces visuais**. In Anais do 1º. Workshop de Realidade Virtual, São Carlos, SP, 1997.

VAN DAM, S. A. Post-wimp user interfaces. **Communications of ACM,** Vol. 40. No. 2, February 1997.

ANEXO A

```
create table estado (
        cd_uf         number(4,0)       not null ,
        ds_uf         varchar2(35)      not null ,
        sg_uf         varchar2(2)       not null ,
        primary key (cd_uf)
) ;
create table cidade (
        cd_cidade     number(8,0)       not null ,
        cd_uf         number(4,0)       not null ,
        ds_cidade     varchar2(40)      not null ,
        nr_ddd        number(4,0),
        primary key (cd_cidade)
) ;
create table endereco (
        cd_ender      number(10,0)      not null ,
        cd_cidade     number(8,0)       not null ,
        nr_cep        varchar2(8)       not null ,
        nr_ender      varchar2(15)      not null ,
        nm_rua        varchar2(30)      not null ,
        cd_tel        number(10,0),
        cd_cli        number(10,0),
        cd_func       number(5,0),
        cd_vend       number(3,0),
```

```
        primary key (cd_ender)
);
create table telefone (
        cd_tel          number(10,0)        not null ,
        nr_ddd          number(3,0)         not null ,
        nr_tel          number(10,0)        not null ,
        ds_email        varchar2(40) ,
        primary key (cd_tel)
) ;
create table cliente (
        cd_cli          number(10,0)        not null ,
        rz_social       varchar2(40)        not null ,
        nm_fantasia     varchar2(40)        not null ,
        nr_cnpj         varchar2(14)        not null ,
        cd_vend         number(3,0)         not null ,
        primary key (cd_cli)
) ;
create table pedido (
        nr_pedido       number(12,0)        not null ,
        dt_pedido       date                not null ,
        st_pedido       varchar2(1)         not null ,
        dt_entrega      date                not null ,
        qt_pedido       number(10,0)        not null ,
        vr_pedido       number(8,2)         not null,
        tp_pagto        char(1)             not null ,
```

```
        cd_cli          number(10,0)        not null ,
        primary key (nr_pedido)
) ;
create table vendedor (
        cd_vend         number(3,0)         not null ,
        nr_cpf          varchar2(14)        not null ,
        nm_vend         varchar2(40)        not null ,
        pc_comissao     number(5,2)         not null ,
        dt_inicio       date                not null ,
        primary key (cd_vend)
) ;
create table ajuda_custo (
        cd_ajuda        number(5,0)         not null ,
        vr_ajuda        number(6,2)         not null ,
        dt_ajuda        date                not null ,
        cd_vend         number(3,0)         not null ,
        primary key (cd_ajuda,cd_vend)
) ;
create table produto (
        cd_prod         number(5,0)         not null ,
        ds_prod         varchar2(40)        not null ,
        qt_alt          number(6,2)         not null ,
        qt_larg         number(6,2)         not null ,
        ds_cor          varchar2(20)        not null ,
        qt_estoq        number(11,2)        not null ,
```

```
        dt_ult_cont     date                not null ,
        primary key (cd_prod)
) ;
create table funcionario (
        cd_func         number(5,0)         not null ,
        nm_func         varchar2(40)        not null ,
        dt_admissao     date                not null ,
        vr_sal          number(8,2)         not null ,
        nr_cpf          varchar2(14)        not null ,
        primary key (cd_func)
) ;
create table parcelas (
        nr_parcela      number(3,0)         not null,
        dt_vencto       date                not null,
        vr_parcela      number(7,2)         not null,
        dt_pagto        date,
        nr_pedido       number(12,0)        not null,
        primary key (nr_parcela,nr_pedido)
) ;
create table pedprod (
        nr_pedido       number(12,0)        not null ,
        cd_prod         number(5,0)         not null ,
        qt_ped          number(10,0)        not null ,
        vr_ped          number(11,2)        not null ,
        primary key (nr_pedido,cd_prod)
```

```
) ;
-- create foreign keys section
alter table cidade add  foreign key (cd_uf) references estado (cd_uf) ;
alter table endereco add  foreign key (cd_cidade) references cidade (cd_cidade) ;
alter table endereco add  foreign key (cd_tel) references telefone (cd_tel) ;
alter table pedido add  foreign key (cd_cli) references cliente (cd_cli) ;
alter table endereco add  foreign key (cd_cli) references cliente (cd_cli) ;
alter table parcelas add  foreign key (nr_pedido) references pedido (nr_pedido) ;
alter table pedprod add  foreign key (nr_pedido) references pedido (nr_pedido) ;
alter table endereco add  foreign key (cd_vend) references vendedor (cd_vend) ;
alter table ajuda_custo add  foreign key (cd_vend) references vendedor (cd_vend) ;
alter table cliente add  foreign key (cd_vend) references vendedor (cd_vend) ;
alter table pedprod add  foreign key (cd_prod) references produto (cd_prod) ;
alter table endereco add  foreign key (cd_func) references funcionario (cd_func) ;
```